تعلم لغة:

p h p

المادة مترجمة من موقع:

W3schools

ترجمة: علي حمدان العمامي

PHP

هي لغة برمجة نصية للخوادم، وأداة قوية لإنشاء صفحات ويب حيوية وتفاعلية، وهي بديل واسع الاستخدام ومجاني وفعال للمنافسين مثل ASP من Microsoft.

PHP 7 هو أحدث إصدار مستقر.

مــثـــال:

```
<!DOCTYPE html>
<html>
<body>

<?php
echo " php بـــ الأول النـصي بـرنـامـجي ";
 ?>

</body>
</html>
```

-----(((مـقـدمـة PHP)))-----

تستخدم أكواد PHP التنفيدية للخوادم.

ما يجب أن تعرفه:

قبل المتابعة، يجب أن يكون لديك فهم أساسي لما يلي:

- HTML
- CSS
- JavaScript

ما هي PHP؟

- ❖ **PHP** هي اختصار لـ "PHP: Hypertext Preprocessor".
- ❖ **PHP** هي لغة برمجة مفتوحة المصدر تستخدم على نطاق واسع.
- ❖ يتم تنفيذ البرامج النصية لـ PHP على الخوادم.
- ❖ **PHP** مجانية للتنزيل والاستخدام.

PHP هي لغة قوية ومشهورة!

أنها قوية بما يكفي لتكون في صميم أكبر نظام تدوين على شبكات الإنترنت (WordPress)!

إنها عميقة بما يكفي لتشغيل أكبر شبكة اجتماعية (Facebook)!

من السهل أيضًا أن تكون لغة الخادم للمبتدئين!

ما هو ملف PHP؟

يمكن أن تحتوي ملفات PHP على أكواد: HTML، CSS، JavaScript، PHP.

يتم تنفيذ أكواد PHP على الخوادم، ويتم إرجاع النتيجة إلى المتصفح كـ HTML عادي.

ملفات PHP لها امتداد ".php".

ماذا يمكن أن تفعل PHP؟

- ❖ إنشاء محتوى حيوي لصفحات الإنترنت
- ❖ إنشاء وفتح وقراءة وكتابة وحذف وإغلاق الملفات على الخوادم
- ❖ جمع بيانات النموذج
- ❖ إرسال واستقبال ملفات تعريف الارتباط
- ❖ إضافة أو حذف أو تعديل البيانات في قاعدة البيانات الخاصة بك
- ❖ التحكم في وصول المستخدم
- ❖ تشفير البيانات
- ❖ لا تقتصر PHP على إخراج صفحات HTML فقط. بل يمكنها إخراج الصور وملفات PDF. يمكنها أيضًا إخراج أي نص، مثل XHTML وXML.

لماذا PHP؟

- ❖ PHP تعمل على منصات متعددة (Windows،Linux،Unix،Mac OS X، إلخ).
- ❖ PHP تتوافق مع جميع الخوادم المستخدمة حاليًا (Apache،IIS... إلخ).
- ❖ PHP تدعم مجموعة واسعة من قواعد البيانات.
- ❖ PHP مجانية. قم بتنزيلها من موقع PHP الرسمي: www.php.net.
- ❖ PHP سهلة التعلم وتعمل بكفاءة من جانب الخادم.

ما الجديد في PHP 7

- ❖ PHP 7 أسرع بكثير من الإصدار الثابت السابق المعروف (PHP 5.6).
- ❖ تدعم PHP 7 تعريفات أكثر صرامة لمدخلات الدوال.
- ❖ تدعم PHP 7 العمليات الجديدة (مثل عملية السفينة الفضاء: <=>).

تثبيت PHP

ماذا أحتاج؟

لبدء استخدام PHP، يمكنك:

- ❖ البحث عن مضيف ويب يدعم PHP وMySQL.
- ❖ قم بتثبيت خادم ويب على حاسوبك، ثم قم بتثبيت PHP و MySQL.

استخدم مضيف ويب يدعم PHP

إذا قام خادمك بتنشيط دعم PHP، فلن تحتاج إلى القيام بأي شيء.

ما عليك سوى إنشاء بعض ملفات php.، ووضعها في مجلد الويب الخاص بك، وسيقوم الخادم تلقائيًا بتحليلها لك.

لا تحتاج إلى ترجمة أي شيء أو تثبيت أي أدوات إضافية.

نظرًا لأن PHP مجانية، تقدم معظم مضيفات الويب دعم PHP.

قم بإعداد PHP على الحاسوب الخاص بك

ومع ذلك، إذا كان خادمك لا يدعم PHP، فيجب عليك:

- ❖ تثبيت خادم الويب.
- ❖ تثبيت PHP.
- ❖ تثبيت قاعدة بيانات، مثل: MySQL.

يحتوي موقع PHP الرسمي (PHP.net) على تعليمات التثبيت:

http://php.net/manual/ar/install.php

بنيـة جمـل PHP

يتم تنفيذ البرنامج النصي لـ PHP على الخوادم، ويتم إرسال نتيجة HTML العادية مرة أخرى إلى المتصفح.

بنية الجملة الأساسية لـ PHP

يمكن وضع نص PHP في أي مكان في المستند.

يبدأ نص PHP بـ <?php وينتهي بـ ?>:

```
<?php
// ضع أكواد php هنا
 ?>
```

الامتداد الافتراضي لملفات PHP هو ".php".

يحتوي ملف PHP عادةً على أوسمة HTML، وبعض التعليمات النصية البرمجية لـ PHP.

لدينا مثال بسيط لملف PHP، مع برنامج PHP نصي يستخدم دالة "echo" لإخراج نص "Hello World!" على صفحة الويب:

مـــثـــال:

```
<!DOCTYPE html>
<html>
<body>

<h1>My first PHP page</h1>

<?php
echo "Hello World!";
?>

</body>
</html>
```

ملاحظة: تنتهي جمل PHP بفاصلة منقوطة.

الحساسية لحالة الأحرف

في PHP، لا توجد كلمات رئيسية حساسة لحالة الأحرف (مثل if وelse وwhile وecho، وغيرها)، والفئات والدوال والدوال المعرفة من قبل المستخدم.

في المثال التالي، جميع عبارات echo الثلاثة متساوية ومسموح بها:

مــثـــال:

```
<!DOCTYPE html>
<html>
<body>

<?php
ECHO "Hello World!<br>";
echo "Hello World!<br>";
EcHo "Hello World!<br>";
?>

</body>
</html>
```

ملاحظة: جميع أسماء المتغيرات حساسة لحالة الأحرف!

انظر للمثال التالي؛ سيتم عرض قيمة الجملة الأولى $color فقط! هذا لأنه يتم التعامل مع $color و$COLOR و$coLOR كثلاثة متغيرات مختلفة:

مـثـــال:

```
<!DOCTYPE html>
<html>
<body>

<?php
$color = "red";
echo "My car is " . $color . "<br>";
echo "My house is " . $COLOR . "<br>";
echo "My boat is " . $coLOR . "<br>";
?>

</body>
</html>
```

تعليقات PHP

التعليق في أكواد PHP هو سطر لا يتم تنفيذه كجزء من البرنامج. الغرض الوحيد منه هو توضيح للشخص الذي ينظر إلى الكود.

يمكن استخدام التعليقات من أجل:

- ❖ جعل الكود مفهوما لمن يقرأه.
- ❖ ذكّر نفسك بما فعلته – عانى معظم المبرمجين من العودة إلى عملهم بعد عام أو عامين ولابد من إعادة فهم ما كتبوه! يمكن أن تذكرك التعليقات بما كنت تفكر فيه عندما كتبت الكود.

PHP تدعم عدة طرق للتعليق:

مـثـــال:

بنية التعليقات ذات السطر الواحد:

```
<?php
// هذا تعليق سطر واحد
# هذا أيضا تعليق سطر واحد
?>
```

مــثـــال:

بنية التعليقات متعدد الأسطر:

```
<?php
/*
هذا تعليق متعدد الأسطر
*/
?>
```

مــثـــال:

استخدام التعليقات لإهمال جزء من الكود:

```
<?php
// يمكنك استخدام التعليقات لإهمال أجزاء من الكود
$x = 5 /* + 15 */ + 5;
echo $x;
?>
```

المـتـغيرات

المتغيرات هي "حاويات" لتخزين المعلومات.

في PHP، يبدأ المتغير بعلامة $، متبوعًا باسم المتغير:

مــثـــال:

```
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

بعد تنفيذ الجمل السابقة، سيحتفظ المتغير `$txt` بالقيمة `Hello world!` وسيحتفظ المتغير `$x` بالقيمة `5`، وسيحتفظ المتغير `$y` بالقيمة `10.5`.

ملاحظات:-

عندما تقوم بتعيين قيمة نصية لمتغير، ضع علامات اقتباس حول القيمة.

على عكس لغات البرمجة الأخرى، ليس لدى PHP أمر لإعلان المتغير. يتم إنشاؤه في اللحظة التي تحدد فيها القيمة لأول مرة.

يمكن أن يكون للمتغير اسم قصير (مثل x و y) أو اسم وصفي، مثل: (age، carname، total_volume).

قواعد متغيرات PHP:

- يبدأ المتغير بعلامة $، متبوعًا باسم المتغير.
- يجب أن يبدأ اسم المتغير بحرف أو شرطة سفلية (_).
- لا يمكن أن يبدأ اسم المتغير برقم.
- أسماء المتغيرات يمكن أن تحتوي فقط على أحرف أبجدية ورقمية وشرطات سفلية (A-z و 0-9 و _). /* **يمكنك التسمية بالحروف العربية أيضا***/
- أسماء المتغيرات حساسة لحالة الأحرف ($age و$AGE هما متغيرين مختلفين).

طباعة المتغيرات

غالبًا ما تستخدم جملة echo لإخراج البيانات على الشاشة.

يوضح المثال التالي كيفية إخراج النص والمتغير:

مــثـــال:

```
<?php
$txt = "W3Schools.com";
echo "I love $txt!";
?>
```

المثال التالي سينتج نفس الناتج كما في المثال السابق:

مــثـــال:

```
<?php
$txt = "W3Schools.com";
echo "I love " . $txt . "!";
?>
```

/*عند وضع المتغير بين علامتي تنصيص مزدوجة تطبع قيمته، أما بين علامتي تنصيص مفردة فيطبع اسم المتغير*/

المثال التالي سيخرج مجموع المتغيرين:

مــثـــال:

```
<?php
$x = 5;
$y = 4;
echo $x + $y;
?>
```

ملاحظة: سوف تتعلم المزيد عن جملة echo وكيفية إخراج البيانات على الشاشة في الفصل التالي.

في المثال السابق، لاحظ أننا لم نضطر إلى إخبار PHP بنوع المتغير.

يقوم PHP تلقائيًا بربط نوع البيانات بالمتغير، اعتمادًا على قيمته. نظرًا لعدم تعيين أنواع المتغيرات بالمعنى الدقيق للكلمة، يمكنك القيام بأشياء مثل إضافة سلسلة إلى عدد صحيح دون التسبب في خطأ.

في PHP 7، تمت إضافة تعريفات النوع. يوفر هذا خيارًا لتحديد نوع البيانات المتوقع عند الإعلان عن الدالة، وعن طريق تمكين طلب التدقيق (`strict requirement`)، سيؤدي ذلك إلى "خطأ فادح" في عدم تطابق النوع.

ملاحظة: سوف تتعلم المزيد عن طلب `strict` و`non-strict` أو التدقيق و عدم التدقيق، وتعريفات نوع البيانات في فصل دوال PHP.

نطاق متغيرات PHP

في PHP، يمكن الإعلان عن المتغيرات في أي مكان في البرنامج النصي.

نطاق المتغير: هو جزء من البرنامج النصي، ويحدد إين يمكن الرجوع إلى المتغير أو استخدامه.

يحتوي PHP على ثلاثة نطاقات مختلفة:

- ❖ **محلـــي local**
- ❖ **عالمي global**
- ❖ **ثابـــت static**

النطاق العالمي والمحلي

المتغير المعرف خارج الدالة هو في النطاق العالمي، ولا يمكن الوصول إليه إلا خارج الدالة:

مـــثـــال:

متغير النطاق العالمي

```
<?php
$x = 5; // نطاق عالمي

function myTest()
{
    // استخدام x داخل هذه الدالة سينتج خطأ
    echo "<p>Variable x inside function is: $x</p>";
}

myTest();

echo "<p>Variable x outside function is: $x</p>";
?>
```

المتغير الذي تم الإعلان عنه **داخل** الدالة هو في نطاق محلي ولا يمكن الوصول إليه إلا داخل هذه الدالة:

مـــثــــال:

متغير النطاق المحلي

```
<?php
function myTest()
{
    $x = 5; // نطاق محلي
    echo "<p>Variable x inside function is: $x</p>";
}
myTest();

// استخدام x خارج هذه الدالة سينتج خطأ
echo "<p>Variable x outside function is: $x</p>";
?>
```

يمكن أن يكون لديك متغيرات محلية بنفس الاسم في دوال مختلفة، لأن المتغيرات المحلية يتم التعرف عليها فقط داخل الدوال التي تم الإعلان فيها.

الكلمة الأساسية (العالمي "The global")

يتم استخدام الكلمة الأساسية `global` للوصول إلى متغير عالمي داخل دالة.

للقيام بذلك، استخدم الكلمة الأساسية `global` قبل المتغيرات (داخل الدالة):

مـثـــال:

```
<?php
$x = 5;
$y = 10;

function myTest() {
    global $x, $y;
    $y = $x + $y;
}

myTest();
echo $y; // 15 المخرج
?>
```

تقوم PHP أيضًا بتخزين جميع المتغيرات العالمية في مصفوفة تسمى `$GLOBALS[index]`.

الـ *index* يحمل اسم المتغير. يمكن الوصول إلى هذه المجموعة أيضًا من داخل الدوال ويمكن استخدامها لتحديث المتغيرات العالمية مباشرةً.

يمكن إعادة كتابة المثال السابق كما يلي:

مـثـــال:

```
<?php
$x = 5;
$y = 10;

function myTest()
{
    $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}

myTest();
echo $y; // 15 المخرج
?>
```

الكلمة الأساسية (الثابت "The static")

عادة، عندما يتم الانتهاء من دالة أو تنفيذها، يتم حذف جميع المتغيرات. ومع ذلك، في بعض الأحيان نريد متغير محلي لا يحذف. نحن بحاجة إليه للحصول على دالة أخرى.

للقيام بذلك، نستخدم الكلمة الأساسية static عندما نقوم بتعريف المتغير لأول مرة:

مـــثـــال:

```
<?php
function myTest() {
    static $x = 0;
    echo $x;
    $x++;
}

myTest();
myTest();
myTest();
?>
```

بعد ذلك، في كل مرة يتم فيها استدعاء الدالة، سيظل هذا المتغير محتويا على المعلومات التي احتواها من آخر مرة تم فيها استدعاء الدالة.

ملاحظة: لا يزال المتغير محليًا في الدالة.

جملتي echo وprint

هناك طريقتان أساسيتان للإخراج: echo وprint.

في هذه الدروس التعليمية نستخدم echo أو print في كل مثال تقريبًا. لذلك، يحتوي هذا الفصل على مزيد من المعلومات حول جملتي الإخراج هاتين.

echo وprint هما نفس الشيء. كلاهما يستخدم لإخراج البيانات على الشاشة.

الاختلافات:

- echo لا ترجع قيمة، بينما print ترجع قيمة 1 بحيث يمكن استخدامها في التوضيحات.
- يمكن أن تحتوي echo على معلمات متعددة (على الرغم من أن هذا الاستخدام نادر الحدوث) بينما تأخذ print معلم واحد.
- echo أسرع من print.

جملة echo

يمكن استخدام جملة echo مع أو بدون أقواس: echo أو echo().

- **لعرض النص**

يوضح المثال التالي كيفية إخراج النص باستخدام أمر echo (لاحظ أن النص يمكن أن يحتوي على أوسمة HTML):

مـثـــال:

```
<?php
echo "<h2>PHP is Fun!</h2>";
echo "Hello world!<br>";
echo "I'm about to learn PHP!<br>";
echo "هذا ", "النص ", "مكون ", "بأكثر من معلم.";
?>
```

- **لعرض قيم المتغيرات**

يوضح المثال التالي كيفية إخراج النص والمتغيرات باستخدام جملة echo:

مــثـــال:

```
<?php
$txt1 = "Learn PHP";
$txt2 = "W3Schools.com";
$x = 5;
$y = 4;
echo "<h2>" . $txt1 . "</h2>";
echo "Study PHP at " . $txt2 . "<br>";
echo $x + $y;
?>
```

جملة print

يمكن استخدام جملة print مع أو بدون أقواس: print أو print().

- **لعرض النص**

يوضح المثال التالي كيفية إخراج النص باستخدام أمر print (لاحظ أن النص يمكن أن يحتوي على أوسمة HTML):

مـثـــال:

```
<?php
print "<h2>الـــ php ممتعة!</h2>";
print "مرحبا بالعالم!<br>";
print "أنا أتعلم لغة php";
?>
```

- **لعرض المتغيرات**

يوضح المثال التالي كيفية إخراج النص والمتغيرات باستخدام جملة print:

مــثـــال:

```
<?php
$txt1 = "تـعـلـم php";
$txt2 = "W3Schools.com";
$x = 5;
$y = 4;

print "<h2>" . $txt1 . "</h2>";
print "أتـعـلـم مـن: " . $txt2 . "<br>";
print $x + $y;
?>
```

أنـواع بيـانـات PHP

يمكن للمتغيرات تخزين البيانات من أنواع مختلفة، وأنواع البيانات المختلفة يمكن أن تفعل أشياء مختلفة.

تدعم PHP أنواع البيانات التالية:

- ❖ سلسلة نصية "String"
- ❖ عدد صحيح "Integer"
- ❖ عدد كسري (ويسمى أيضًا مزدوج) "Float & double"
- ❖ منطقية "Boolean"
- ❖ مصفوفة "Array"
- ❖ كائن "Object"
- ❖ لا شيء "NULL"
- ❖ مورد "Resource"

السلسلة "String"

السلسلة عبارة عن سلسلة من الأحرف، مثل "Hello world!".

في السلسلة يمكنك وضع القيمة بين علامتي تنصيص مزدوجة أو فردية:

مـــثـــال:

```
<?php
$x = "مرحبا بالعالم";
$y = 'مرحبا بالعالم';

echo $x;
echo "<br>";
echo $y;
?>
```

الأعداد الصحيحة

نوع بيانات العدد الصحيح هو رقم غير عشري بين 2,147,483,648− و 2,147,483,647.

قواعد الأعداد الصحيحة:

- يجب أن يحتوي على رقم واحد على الأقل.
- يجب ألا يحتوي على علامة عشرية.
- يمكن أن يكون موجبا أو سالبا.
- يمكن اختيار الأعداد الصحيحة بـ: العشري (المستند على 10)، السادس عشري (المستند على 16)، الثماني (المستند على 8)، أو الترميز الثنائي (المستند على 2).

في المثال التالي، $x هو عدد صحيح. ترجع الدالة **var_dump()** نوع البيانات والقيمة:

مــثـــال:

```
<?php
$x = 5985;
var_dump($x); // int(5985) المخرج
?>
```

الأعداد الكسرية

الكسري: هو رقم ذو علامة عشرية أو رقم في الشكل الأسي.

في المثال التالي `$x` هو عدد كسري. ترجع الدالة **`var_dump()`** نوع البيانات والقيمة:

مـــثـــال:

```
<?php
$x = 10.365;
var_dump($x);
?>
```

المنطقية

تمثل المنطقية حالتين محتملتين: TRUE أو FALSE.

```
$x = true;
$y = false;
```

غالبا ما تستخدم المنطقية في الاختبار الشرطي.

المصفوفات

تخزن المصفوفة قيم متعددة في متغير واحد.

في المثال التالي `$cars` عبارة عن مصفوفة. ترجع الدالة `var_dump()` نوع البيانات والقيمة:

مــثـــال:

```
<?php
$cars = array("Volvo","BMW","Toyota");
var_dump($cars);
?>
```

الكائن

الكائن: هو نوع البيانات الذي يخزن البيانات والمعلومات حول كيفية معالجة تلك البيانات.

في PHP، يجب الإعلان عن الكائن.

أولا يجب أن نعلن صنف الكائن. لهذا، نستخدم الكلمة الأساسية الفئة "class".

الفئة (class): هي بنية يمكن أن تحتوي على خصائص وطرق:

مــثـــال:

```
<?php
class Car {
    function Car() {
        $this->model = "VW";
    }
}

// إنشاء كائن
$herbie = new Car();

// عرض خاصيات الكائن
echo $herbie->model;
?>
```

قيمة فارغة null

Null: هي نوع بيانات خاصة يمكن أن تحتوي على قيمة واحدة فقط: NULL.

متغير نوع البيانات NULL هو متغير ليس له قيمة مخصصة له.

نصيحة: إذا تم إنشاء متغير بدون قيمة، فإنه يتم تلقائيًا تعيينه كقيمة فارغة.

يمكن أيضًا إفراغ المتغيرات عن طريق تعيين القيمة إلى NULL:

مـــثـــال:

```
<?php
$x = "Hello world!";
$x = null;
var_dump($x);
?>
```

الموارد

نوع المورد خاص ليس نوع بيانات فعلي. إنه تخزين مرجع أو إشارة للدوال والموارد الخارجية لـ PHP.

مثال شائع لاستخدام نوع بيانات المورد، هو: استدعاء قاعدة بيانات.

لن نتحدث عن نوع المورد هنا، لأنه موضوع متقدم.

السـلا سل النصية لـPHP

دوال السلسلة: في هذا الفصل، سنلقي نظرة على بعض الدوال شائعة الاستخدام لمعالجة السلاسل:

strlen() - إرجاع طول السلسلة

مــثـــال:

ارجاع طول السلسلة "Hello world!":

```
<?php
echo strlen("Hello world!"); // 12 المخرج
?>
```

str_word_count() - حساب عدد الكلمات في السلسلة.

مــثـــال:

حساب عدد الكلمات في السلسلة "Hello world!":

```
<?php
echo str_word_count("Hello world!"); // 2
?>
```

strrev() - عكس السلسلة

مــثـــال:

عكس سلسلة "Hello world!":

```
<?php
echo strrev("Hello world!"); //!dlrow olleH المخرج
?>
```

strpos() - البحث عن نص داخل سلسلة

تبحث دالة strpos() عن نص معين داخل سلسلة. إذا تم العثور على مطابق، تقوم الدالة بإرجاع موضع أول حرف من المطابقة. إذا لم يتم العثور على تطابق، فسيتم إرجاع FALSE.

مــثـــال:

عكس السلسلة "Hello world!":

```
<?php
echo strpos("Hello world!", "world"); // outputs 6
?>
```

ملاحظة: الحرف الأول في السلسلة هو 0 (وليس 1).

str_replace() - استبدال نص داخل سلسلة

تقوم دالة str_replace() باستبدال بعض الأحرف ببعض الأحرف الأخرى في السلسلة.

مــثـــال:

استبدال كلمة "world" بـ "Dolly":

```
<?php
echo str_replace("world", "Dolly", "Hello
world!"); // outputs Hello Dolly!
?>
```

الأرقـام

في هذا الفصل، سننظر بعمق في الأعداد الصحيحة، والكسرية، وسلاسل الأرقام.

أرقام PHP

PHP تحول نوع البيانات تلقائيا.

لذلك، إذا قمت بتعيين قيمة عدد صحيح لمتغير، فسيكون نوع هذا المتغير تلقائيًا عددًا صحيحًا. بعد ذلك، إذا قمت بتعيين سلسلة لنفس المتغير، فإن النوع سوف يتغير إلى سلسلة.

هذا التحويل التلقائي يمكنه في بعض الأحيان كسر الكود الخاص بك.

الأعداد الصحيحة Intergers

العدد الصحيح: هو رقم بدون أي جزء عشري.

2, 256, -256, 10358, -179567 كلها أعداد صحيحة، بينما 7.56, 10.0, 150.67 كسرية أو عشرية.

لذلك، نوع البيانات عدد-صحيح هو رقم غير عشري بين 2147483648- و 2147483647. سيتم تخزين قيمة أكبر (أو أقل) من هذا، كما هو الكسري، لأنه يتجاوز الحد الأقصى للعدد الصحيح.

شيء مهم آخر يجب معرفته هو أنه حتى إذا كان ناتج عملية `2.5 * 4` هو `10`، فإن النتيجة يتم تخزينها بشكل كسري، لأن أحد المعاملات كان كسري (`2.5`).

فيما يلي بعض القواعد للأعداد الصحيحة:

- يجب أن يحتوي على رقم واحد على الأقل.
- يجب ألا يحتوي على علامة عشرية.
- يمكن أن يكون إيجابيا أو سلبيا.
- يمكن تحديد الأعداد الصحيحة في ثلاثة تنسيقات: عشري (يستند إلى 10) أو ست عشري (يستند إلى 16 – مسبوق بـ0x) أو ثماني (يستند إلى 8 – مسبوق بـ 0)

PHP لديها الدوال التالية للتحقق مما إذا كان نوع المتغير هو عدد صحيح:

- is_int()
- is_integer() - alias of is_int()
- is_long() - alias of is_int()

مـثـــال:

تحقق مما إذا كان نوع المتغير عددًا صحيحًا:

```
<?php
$x = 5985;
var_dump(is_int($x));

$x = 59.85;
var_dump(is_int($x));
?>
```

الأرقام الكسرية

الكسري: هو رقم بالعلامة العشرية أو رقم في شكل أسي.

2.0, 256.4, 10.358, 7.64E+5, 5.56E-5 كلها كسرية.

يمكن لنوع البيانات الكسري عادةً تخزين قيمة تصل إلى 1.7976931348623E+308 (تعتمد على النظام الأساسي)، ويبلغ الحد الأقصى 14 رقم.

يحتوي PHP على الدوال التالية للتحقق مما إذا كان نوع المتغير كسري:

- is_float()
- is_double() - alias of is_float()

مـثــال:

تحقق مما إذا كان نوع المتغير كسري:

```
<?php
$x = 10.365;
var_dump(is_float($x));
?>
```

ما لا نهاية

تعتبر القيمة الرقمية أكبر من PHP_FLOAT_MAX غير محدودة.

تحتوي PHP على الدوال التالية للتحقق مما إذا كانت القيمة الرقمية محدودة أو غير محدودة:

- is_finite()
- is_infinite()

ومع ذلك، ترجع الدالة `var_dump()` نوع البيانات والقيمة:

مــثـــال:

تحقق مما إذا كانت القيمة الرقمية محدودة أم لا نهائية:

```
<?php
$x = 1.9e411;
var_dump($x);
?>
```

NaN

NaN هي اختصار لـ "ليس رقما" "Not a Number".

تستخدم للعمليات الرياضية المستحيلة.

الدالة التالية للتحقق مما إذا كانت القيمة ليست رقمًا:

- is_nan()

ومع ذلك، ترجع الدالة var_dump() نوع البيانات والقيمة:

مــثـــال:

العلمية الغير صحيحة تخرج قيمة NaN:

```
<?php
$x = acos(8);
var_dump($x);
?>
```

السلاسل العددية

يمكن استخدام دالة `is_numeric()` لمعرفة ما إذا كان المتغير رقما. تُرجع الدالة `true` إذا كان المتغير رقمًا أو سلسلة رقمية، و `false` لغير ذلك.

مــثـــال:

تحقق مما إذا كان المتغير رقميا:

```
<?php
$x = 5985;
var_dump(is_numeric($x));

$x = "5985";
var_dump(is_numeric($x));

$x = "59.85" + 100;
var_dump(is_numeric($x));

$x = "Hello";
var_dump(is_numeric($x));
?>
```

ملاحظة: من PHP 7.0: تقوم دالة `is_numeric()` بإرجاع `FALSE` للسلاسل الرقمية بالشكل السادس عشري (مثل 0xf4c3b00c)، حيث لم تعد تعتبر سلاسل رقمية.

ادخال سلاسل نصية وكسور على الأعداد الصحيحة

في بعض الأحيان تحتاج إلى وضع قيمة عددية في نوع بيانات آخر.

غالبًا ما تستخدم الدالة (int) أو (integer) أو intval() لتحويل القيمة إلى عدد صحيح.

مـثــال:

وضع كسور وسلاسل في العدد الصحيح:

```
<?php
// Cast float to int
$x = 23465.768;
$int_cast = (int)$x;
echo $int_cast;

echo "<br>";

// Cast string to int
$x = "23465.768";
$int_cast = (int)$x;
echo $int_cast;
?>
```

الثوابت

الثوابت تشبه المتغيرات، إلا أنه بمجرد تعريفها، لا يمكن تغييرها أو عدم تعريفها.

الثابت هو معرف (اسم) لقيمة بسيطة. لا يمكن تغيير القيمة داخل البرنامج النصي.

يبدأ اسم الثابت بحرف أو شرطة سفلية (بدون علامة $ قبل الاسم الثابت).

> **ملاحظة:** على عكس المتغيرات، تكون الثوابت عالمية بشكل عام عبر البرنامج النصي بأكمله.

إنشاء ثابت

لإنشاء ثابت، استخدم الدالة **define()**.

بنية الجملة

```
define (الاسم, القيمة, غير حساسة لحالة الأحرف)
```

المعلمات:

- **الاسم:** اسم الثابت.
- **القيمة:** قيمة الثابت.
- **غير حساس لحالة الأحرف:** لتحديد ما إذا كان يجب أن يكون الاسم الثابت غير حساس لحالة الأحرف. الافتراضي `false`.

مــثـــال:

إنشاء ثابت باسم **حساس لحالة الأحرف:**

```
<?php
define("GREETING", "Welcome to W3Schools.com!");
echo GREETING;
?>
```

مــثـــال:

إنشاء ثابت باسم **غير حساس لحالة الأحرف:**

```
<?php
define("GREETING", "Welcome to W3Schools.com!", true);
echo greeting;
?>
```

مصفوفات ثابتة

في PHP7، يمكنك إنشاء مصفوفة ثابتة باستخدام دالة **define()**.

مـثــال:

إنشاء مصفوفة ثابتة:

```
<?php
define("cars",
[
    "Alfa Romeo",
    "BMW",
    "Toyota"
]);
echo cars[0];
?>
```

ثوابت عالمية

الثوابت هي تلقائيا عالمية يمكن استخدامها عبر البرنامج النصي بأكمله.

مــثـــال:

في هذا المثال ثابت داخل دالة، حتى إذا تم تعريفه خارج الدالة:

```
<?php
define("GREETING", "Welcome to W3Schools.com!");

function myTest() {
    echo GREETING;
}

myTest();
?>
```

العمليات

يتم استخدام العمليات لإجراء عمليات على المتغيرات وقيمها.

تقسم الـ PHP العمليات كالتالي:

- عمليات رياضية.
- عمليات التخصيص
- عمليات المقارنة.
- عمليات الزيادة والنقصان.
- عمليات منطقية.
- عمليات السلسلة.
- عمليات المصفوفة.
- علميات التخصيص الشرطي.

العمليات الرياضية

يتم استخدام العمليات الرياضية مع القيم الرقمية لأداء العمليات الحسابية المعروفة، مثل: الجمع والطرح والضرب إلخ.

العملية	الاسم	مثال
+	جمع	`$x + $y`
-	طرح	`$x - $y`
*	ضرب	`$y * $x`
/	قسمة	`$y / $x`
%	باقي القسمة	`$y % $x`
**	أس	`$y ** $x`

عمليات التخصيص

يتم استخدام عمليات التخصيص لوضع القيم الرقمية في المتغيرات.

عامل التخصيص الأساسي في PHP هو "**=**".

التخصيص	مثل ..	وصف
x = y	x = y	سيتم وضع قيمة اليسار في قيمة اليمين
x += y	x = x + y	جمع
x -= y	x = x - y	طرح
x *= y	x = x * y	ضرب
x /= y	x = x / y	قسمة
x %= y	x = x % y	باقي القسمة

علميات المقارنة

تُستخدم عمليات المقارنة في PHP لمقارنة قيمتين (رقم أو سلسلة):

العملية	الاسم	مثال	الوصف
==	يساوي	$x == $y	سيرجع true إذا كان x يساوي y
===	يطابق	$x === $y	سيرجع true إذا كان x يساوي y، وكانوا من نفس النوع
!=	لا يساوي	$x != $y	سيرجع true إذا كان x لا يساوي y
<>	لا يساوي	$x <> $y	سيرجع true إذا كان x لا يساوي y
!==	لا يطابق	$x !== $y	سيرجع true إذا كان x لا يساوي y، أو لم يكونوا من نفس النوع
>	أكبر من	$x > $y	سيرجع true إذا كان x أكبر من y
<	أصغر من	$x < $y	سيرجع true إذا كان x أصغر منy
>=	أكبر من أو يساوي	$x >= $y	سيرجع true إذا كان x أكبر من أو يساوي y
<=	أصغر من أو يساوي	$x <= $y	سيرجع true إذا كان x أصغر من أو يساوي y
<=>	Spaceship	$x <=> $y	سيرجع رقم صحيح أقل من أو يساوي أو أكبر من الصفر، وهذا يتوقف على ما إذا كان $x أقل من أو يساوي أو أكبر من $y. قدم في PHP 7.

عمليات الزيادة والنقصان

يتم استخدام عمليات الزيادة لزيادة قيمة المتغير.

وتستخدم عمليات النقصان لتقليل قيمة المتغير.

العملية	الاسم	الوصف
++$x	الزيادة أولا	يزيد $x واحد ثم يرجع قيمة $x
$x++	الزيادة بعد	يرجع $x ثم يزيد واحد
--$x	النقص أولا	ينقص $x واحد، ثم يرجع قيمته
$x--	النقص بعد	يرجع قيمة $x ثم يزيدها واحد

العمليات المنطقية

يتم استخدام العمليات المنطقية لـ PHP لدمج الجمل الشرطية.

العملية	الاسم	مثال	الوصف
and	و	$x and $y	صحيح إذا كان $x و $y صحيحان
or	أو	$x or $y	صحيح إذا كان $x أو $y صحيح
xor		$x xor $y	صحيح إذا كان $x أو $y صحيحان، ولكن ليس كلاهما
&&	و	$x && $y	صحيح إذا كان $x و $y صحيحان
||	أو	$x || $y	صحيح إذا كان $x أو $y صحيح
!	ليس	!$x	صحيح إذا كان $x ليس صحيح

عمليات السلاسل النصية

تحتوي PHP على عاملين تم تصميمهما خصيصًا للسلاسل النصية.

العملية	الاسم	مثال	الوصف
`.`	تجميع	`$text1 . $text2`	جمع `$text1` و `$text2` (في الإخراج)
`.=`		`$text1 .= $txt2`	إضافة `$text2` لقيمة `$text1`

عمليات المصفوفة

يتم استخدام عمليات المصفوفة لمقارنة المصفوفات:

العملية	الاسم	مثال	الوصف
`+`	توحيد	`$x + $y`	توحيد `$x` و `$y`
`==`	مساواة	`$x == $y`	يرجع صح إذا كان `$x` و `$y` لهما نفس أزواج المفتاح/القيمة
`===`	مطابقة	`$x === $y`	يرجع صح إذا كان `$x` و `$y` لهما نفس أزواج المفاتيح/القيمة في نفس الترتيب ومن نفس النوع
`!=`	عدم المساواة	`$x != $y`	يرجع صح إذا كان `$x` لا يساوي `$y`
`<>`	عدم المساواة	`$x <> $y`	يرجع صح إذا كان `$x` لا يساوي `$y`
`!==`	عدم المطابقة	`$x !== $y`	يرجع صح إذا كان `$x` لا يطابق `$y`

عمليات التخصيص الشرطي

يتم استخدام عمليات التخصيص الشرطي لتخصيص قيمة وفقًا لشروط:

العملية	الاسم	مثال	الوصف
?:	ثلاثي "Ternary"	`$x = expr1 ? expr2 : expr3`	يرجع قيمة $x. قيمة $x هي expr2 إذا كانت expr1=TRUE. قيمة $x هي expr3 إذا كان expr1 = FALSE
??	Null coalescing	`$x = expr1 ?? expr2`	يرجع قيمة $x. قيمة $x هي expr1 في حالة وجود expr1، وهي ليست فارغة. إذا كان expr1 غير موجود أو كان NULL، فإن قيمة $x هي expr2. قدم في PHP 7

جمل if ... else ... elesif

تستخدم الجمل الشرطية لتنفيذ إجراءات مختلفة بناءً على شروط معينة.

في كثير من الأحيان عند كتابة التعليمات البرمجية، تريد تنفيذ إجراءات مختلفة لظروف معينة. يمكنك استخدام الجمل الشرطية في التعليمات البرمجية الخاصة بك للقيام بذلك.

في PHP لدينا العبارات الشرطية التالية:

- ❖ جملة `if` – ينفذ بعض الكود إذا كان هناك شرط واحد صحيح.
- ❖ جملة `if ... else` – ينفذ بعض الكود إذا كان الشرط صحيحا وكود آخر إذا كان هذا الشرط خاطئًا.
- ❖ جملة `if...elseif...else` – ينفذ الأكواد المختلفة لأكثر من شرطين.
- ❖ جملة `switch` – تختار واحدة من كتل الكود العديدة التي سيتم تنفيذها.

جملة if

جملة `if` تنفذ بعض التعليمات البرمجية إذا كان هناك شرط واحد صحيح.

بنية الجملة

```
if (الشرط ) {
    الكود المراد تشغيله إذا كان الشرط صحيح;
}
```

مـثـــال:

إخراج "Have a good day!" إذا كان الوقت الحالي (الساعة) أقل من ٢٠:

```
<?php
$t = date("H");

if ($t < "20") {
    echo "Have a good day!";
}
?>
```

جملة الـ if ... else

الجملة if ... else تنفذ بعض الأكواد إذا كان الشرط صحيحًا وكود آخر إذا كان هذا الشرط خاطئًا.

بنية الجملة

```
if (الشرط ) {
    الكود المراد تشغيله إذا كان الشرط صحيحا;
} else {
    الكود المراد تشغيله إذا كان الشرط خاطئا;
}
```

مــثـــال:

إخراج "Have a good day!" إذا كان الوقت الحالي أقل من 20، و "Have a good night!" لغير ذلك:

```
<?php
$t = date("H");

if ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
```

جملة if … elseif … else

بنية الجملة:

```
if (الشرط ) {
    الكود المراد تشغيله إذا كان الشرط صحيحا;
} elseif (الشرط ) {
     الكود المراد تشغيله إذا كان الشرط الأول خطأ
وهذا الشرط صحيح;
} else {
    الكود المراد تشغيله إذا كانت الشروط كلها خطأ;
}
```

مــثـــال:

إخراج "Have a good day!" إذا كان الوقت الحالي أقل من 10، و "Have a good day!" إذا كان الوقت الحالي أقل من 20. غير ذلك سيخرج "Have a good night!":

```
<?php
$t = date("H");
if ($t < "10") {
    echo "Have a good morning!";
} elseif ($t < "20") {
    echo "Have a good day!";
} else {
    echo "Have a good night!";
}
?>
```

جملة التبديل switch

يتم استخدام جملة `switch` لأداء إجراءات مختلفة بناءً على ظروف مختلفة.

استخدم جملة `switch` لاختيار أحد كتل التعليمات البرمجية العديدة المراد تنفيذها.

بناء الجملة

```
switch (n) {
    case label1:
        code to be executed if n=label1;
        break;
    case label2:
        code to be executed if n=label2;
        break;
    case label3:
        code to be executed if n=label3;
        break;
    ...
    default:
        code to be executed if n is different from
all labels;
}
```

هذه هي الطريقة التي تعمل بها: أولاً، لدينا تعبير واحد *n* (غالبًا يكون متغير)، يتم تقييمه مرة واحدة. ثم تتم مقارنة قيمة التعبير مع القيم لكل حالة في البنية. إذا كان هناك تطابق، يتم تنفيذ كتلة التعليمات البرمجية المرتبطة بهذه الحالة. استخدم `break` لمنع تشغيل الكود في الحالة التالية تلقائيًا. يتم استخدام الجملة `default` إذا لم يتم العثور على تطابق.

مـــثـــال:

إخراج "Have a good day!" إذا كان الوقت الحالي أقل من 10، و "Have a good day!" إذا كان الوقت الحالي أقل من 20. غير ذلك سيخرج "Have a good night!":

```
<?php
$favcolor = "red";

switch ($favcolor) {
    case "red":
        echo "Your favorite color is red!";
        break;
    case "blue":
        echo "Your favorite color is blue!";
        break;
    case "green":
        echo "Your favorite color is
green!";
        break;
    default:
        echo "Your favorite color is neither
red, blue, nor green!";
}
?>
```

الحـلـقـات

في الفصول التالية سوف تتعلم كيفية تكرار التعليمات البرمجية باستخدام حلقات PHP.

حلقات PHP

في كثير من الأحيان عند كتابة التعليمات البرمجية، تريد نفس كتلة التعليمات البرمجية مرارا وتكرارا لعدد معين من المرات. لذلك، بدلاً من إضافة العديد من أسطر التعليمات البرمجية متساوية تقريبًا في البرنامج النصي، يمكننا استخدام الحلقات.

يتم استخدام الحلقات لتنفيذ نفس الكود من التعليمات البرمجية مرارًا وتكرارًا، طالما كان هناك شرط معين صحيح.

في PHP، لدينا أنواع الحلقة التالية:

- **while**
- **do ... while**
- **for**
- **foreach** – تكرار كتلة أكواد لكل عنصر في المصفوفة.

في الفصول التالية سوف نشرح ونعطي أمثلة على كل نوع.

الحلقة while – تكرار كتلة أكواد طالما أن الشرط المحدد صحيح.

بنية الجملة

```
while (condition is true) {
    code to be executed;
}
```

يعرض المثال التالي الأرقام من 1 إلى 5:

مـثـــال:

```
<?php
$x = 1;

while($x <= 5) {
    echo "The number is: $x <br>";
    $x++;
}
?>
```

شرح المثال:

- ❖ `$x = 1;` – ضبط عداد الحلقة، اضبط قيمة البداية على 1.
- ❖ `$x <= 5` – استمر في الحلقة طالما أن `$x` أقل من أو تساوي 5.
- ❖ `$x++;` – قم بزيادة قيمة عداد الحلقة بمقدار ١ لكل تكرار.

يصل هذا المثال إلى 100 بواسطة عشرات:

مــثـــال:

```
<?php
$x = 0;

while($x <= 100) {
    echo "The number is: $x <br>";
    $x+=10;
}
?>
```

شرح المثال:

- ❖ `$x = 0;` - ضبط عداد الحلقة (`$x`)، وتعيين قيمة البداية إلى 0.
- ❖ `$x <= 100` - استمر في الحلقة طالما `$x` أقل من أو يساوي 100.
- ❖ `$x+=10;` - زيادة عداد الحلقة بقيمة 10 لكل تكرار.

حلقة do while loop

حلقة `do ... while` - يتم تنفيذ حلقة من التعليمات البرمجية مرة واحدة، ثم يكرر الحلقة طالما كان الشرط المحدد صحيحًا.

بنية الجملة

```
do {
    الكود الذي سينفذ;
} while (الشرط صحيح);
```

أمثلة:

المثال أدناه يعين أولاً متغير `$x` إلى 1 (`$x=1`). بعد ذلك، ستكتب حلقة التكرار بعض المخرجات، ثم تزيد المتغير `$x` بـ 1. ثم يتم التحقق من الشرط (يكون `$x` أقل من، أو يساوي 5؟)، وستستمر الحلقة في التشغيل طالما `$x` أقل من أو يساوي 5:

مـثـــال:

```
<?php
$x = 1;

do {
    echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
?>
```

ملاحظة: في حلقة `do...while`، يتم اختبار الشرط بعد تنفيذ الجمل داخل الحلقة. هذا يعني أن حلقة `do...while` ستنفذ الجملة مرة واحدة على الأقل، حتى إذا كان الشرط خاطئ. انظر المثال التالي.

يعين هذا المثال المتغير `$x` على 6، ثم يقوم بتشغيل الحلقة، ثم يتم التحقق من الشرط:

مـثـــال:

```
<?php
$x = 6;

do {
    echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
?>
```

حلقة for

بناء الجملة

```
for (القيمة الإبتدائية; اختبار القيمة; الزيادة أو النقصان) {
    الكود المنفذ لكل دورة;
}
```

المعلمات:

- ❖ **القيمة الإبتدائية**
- ❖ **اختبار القيمة:** تقييم لكل تكرار. إذا تم تقييمه إلى TRUE، تستمر الحلقة. إذا تم تقييمه إلى FALSE، تنتهي الحلقة.
- ❖ **الزيادة أو النقصان:** يزيد من قيمة عداد الحلقة أو ينقص.

أمثلة:

يعرض المثال التالي الأرقام من 0 إلى 10:

مــثـــال:

```
<?php
for ($x = 0; $x <= 10; $x++) {
    echo "The number is: $x <br>";
}
?>
```

شرح المثال:

- ❖ `$x = 0;` – ضبط عداد الحلقة (`$x`)، وتعيين قيمة البداية إلى 0.
- ❖ `$x <= 10;` – استمرار الحلقة طالما `$x` أقل من أو يساوي 10.
- ❖ `$x++` – زيادة عداد الحلقة بعد كل تكرار.

مــثـــال:

```
<?php
for ($x = 0; $x <= 10; $x++) {
    echo "The number is: $x <br>";
```

```
}
?>
```

حلقة لكل foreach

حلقة foreach تعمل فقط على المصفوفات، ويتم استخدامها لإجراء حلقة داخل كل زوج مفتاح/قيمة في مصفوفة.

بناء الجملة

```
foreach ($مصفوفة as $قيمة ) {
  الكود الذي سينفذ;
}
```

لكل تكرار حلقة، يتم تعيين قيمة عنصر المصفوفة الحالية إلى $value ويتحرك مؤشر المصفوفة بواحد، حتى يصل إلى عنصر المصفوفة الأخير.

أمثلة

سيقوم المثال التالي بإخراج قيم المصفوفة المحددة ($colors):

مــثـــال:

```
<?php
$colors
= array("red", "green", "blue", "yellow");

foreach ($colors as $value) {
  echo "$value <br>";
}
?>
```

المثال التالي سينتج المفاتيح والقيم الخاصة بالمصفوفة المحددة ($age):

مــثـــال:

```
<?php
$age
= array("Peter"=>"35", "Ben"=>"37", "Joe"=>"4
3");
foreach($age as $x => $val) {
```

```
  echo "$x = $val<br>";
}
?>
```

دوال PHP

القوة الحقيقية لـ PHP تكمن في دوالها.

تحتوي PHP على أكثر من 1000 دالة مدمجة، بالإضافة إلى ذلك، يمكنك إنشاء دوال مخصصة خاصة بك.

دوال PHP المعرفة من قبل المستخدم

إلى جانب دوال PHP المضمنة، من الممكن إنشاء دوالك الخاصة.

- الدالة: عبارة عن مجموعة من الجمل التي يمكن استخدامها بشكل متكرر في البرنامج.
- لن يتم تنفيذ الدالة تلقائيًا عند تحميل الصفحة.
- سيتم تنفيذ الدالة من خلال استدعائها.

إنشاء دالة معرفة من قبل المستخدم في PHP

يبدأ إعلان الدالة المعرفة من قبل المستخدم بكلمة `function`:

بناء الجملة

```
function اسم الدالة () {
    الكود المراد تنفيذه;
}
```

> **ملاحظة:** يجب أن يبدأ اسم الدالة بحرف أو شرطة سفلية. أسماء الدوال ليست حساسة لحالة الأحرف.

نصيحة: أعطِ الدالة اسمًا يعكس وظيفتها!

في المثال السابق، نقوم بإنشاء دالة باسم "writeMsg()". يشير القوس المجعد المفتوح ({) إلى بداية رمز الدالة، ويشير القوس المجعد المغلق (}) إلى نهاية الدالة. مخرجات الدالة "Hello world!". لاستدعاء الدالة، فقط اكتب اسمها متبوعًا بالأقواس ():

مـــثـــال:

```
<?php
function writeMsg() {
    echo "Hello world!";
}

writeMsg(); // استدعاء الدالة
?>
```

مدخلات الدوال

يمكن تمرير المعلومات إلى الدوال من خلال المدخلات. **المدخل** هو: تماما مثل المتغير.

يتم تعريف المدخلات بعد اسم الدالة، داخل الأقواس. يمكنك إضافة العديد من المدخلات كما تريد، والفصل بينها بفاصلة.

المثال التالي له دالة ذات مدخل واحد (`$fname`). عندما يتم استدعاء دالة `familyName()`، فإننا نمرر أيضًا اسمًا (مثل `Jani`)، ويتم استخدام الاسم داخل الدالة، مما يؤدي إلى إخراج عدة أسماء أولى مختلفة، ولكن اسماء أخيرة متساوية:

مـثـــال:

```
<?php
function familyName($fname) {
    echo "$fname Refsnes.<br>";
}

familyName("Jani");
familyName("Hege");
familyName("Stale");
familyName("Kai Jim");
familyName("Borge");
?>
```

المثال التالي له دالة مع مدخلين (`$fname` و`$year`):

مـثـــال:

```
<?php
function familyName($fname, $year) {
    echo "$fname Refsnes. Born in $year
<br>";
}
```

```
familyName("Hege", "1975");
familyName("Stale", "1978");
familyName("Kai Jim", "1983");
?>
```

في المثال السابق، لاحظ أننا لم نضطر إلى إخبار PHP بنوع البيانات.

يقوم PHP تلقائيًا بربط نوع البيانات بالمتغير، اعتمادًا على قيمته. نظرًا لعدم تعيين أنواع البيانات بالمعنى الدقيق للكلمة، يمكنك القيام بأشياء مثل إضافة سلسلة إلى عدد صحيح دون التسبب في خطأ.

في PHP 7، تمت إضافة تعريفات النوع. هذا يعطينا خيارًا لتحديد نوع البيانات المتوقع عند الإعلان عن دالة، ومن خلال إضافة الإعلان (الصارم "strict")، سيؤدي ذلك إلى "خطأ فادح" في حالة عدم تطابق نوع البيانات.

في المثال التالي، نحاول إرسال رقم وسلسلة إلى الدالة دون استخدام التدقيق "strict":

مــثـــال:

```
<?php
function addNumbers(int $a, int $b) {
    return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is NOT enabled "5 days" is
```

```
changed to int(5), and it will return 10
?>
```

لتمكين التدقيق strict، نحتاج إلى كتابة ;declare(strict_types=1). يجب أن يكون هذا في السطر الأول لملف الـ PHP.

في المثال التالي، نحاول إرسال رقم وسلسلة إلى الدالة، ولكننا قمنا بإضافة الإعلان strict:

مـثـــال:

```
<?php declare(strict_types=1); // strict
requirement

function addNumbers(int $a, int $b) {
    return $a + $b;
}
echo addNumbers(5, "5 days");
// since strict is enabled and "5 days" is
not an integer, an error will be thrown
?>
```

الإعلان strict يفرض استخدام الأشياء بالطريقة المقصودة.

مدخلات قيم عشوائية

يوضح المثال التالي كيفية استخدام المدخل العشوائي. إذا قمنا باستدعاء الدالة `setHeight()` بدون مدخلات، فسوف تأخذ القيمة الافتراضية كمدخل:

مــثـــال:

```
<?php declare(strict_types=1); // طلب التدقيق
function setHeight(int $minheight = 50) {
    echo "The height is : $minheight <br>";
}

setHeight(350);
setHeight(); // will use the default value of 50
setHeight(135);
setHeight(80);
?>
```

دوال – إرجاع القيم

للسماح للدالة بإرجاع القيم، استخدم جملة return:

مـــثـــال:

```
<?php declare(strict_types=1); //طلب التدقيق
function sum(int $x, int $y) {
    $z = $x + $y;
    return $z;
}

echo "5 + 10 = " . sum(5, 10) . "<br>";
echo "7 + 13 = " . sum(7, 13) . "<br>";
echo "2 + 4 = " . sum(2, 4);
?>
```

دوال – إرجاع نوع المعرفات

تدعم PHP 7 أيضًا تعريفات النوع لجملة `return`. كما هو الحال مع إعلان النوع لمدخلات الدالة، من خلال تمكين التدقيق، سوف يظهر "خطأ فادح" عند عدم تطابق النوع.

لتحديد نوع لإرجاع الدالة، أضف نقطتين (:) والنوع مباشرة قبل قوس الفتح المجعد ({) عند تعريف الدالة.

في المثال التالي، نحدد نوع الإرجاع للدالة:

مـثـــال:

```
<?php declare(strict_types=1); //تمكين التدقيق
function addNumbers(float $a, float $b) :
float {
    return $a + $b;
}
echo addNumbers(1.2, 5.2);
?>
```

يمكنك تحديد أنواع إرجاع مختلفة عن أنواع المدخلات، ولكن تأكد من أن الإرجاع هو النوع الصحيح:

مـثـــال:

```
<?php declare(strict_types=1); // تمكين التدقيق
function addNumbers(float $a, float $b) :
int {
    return (int)($a + $b);
}
echo addNumbers(1.2, 5.2);
?>
```

المصفوفات

تخزن المصفوفة قيم متعددة في متغير واحد:

مــثـــال:

```
<?php
$cars      = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>
```

ماهي المصفوفة؟

المصفوفة هي متغير خاص، يمكنها أن تحمل أكثر من قيمة في وقت واحد.

إذا كان لديك قائمة بالعناصر (قائمة بأسماء السيارات، على سبيل المثال)، فقد يبدو تخزين السيارات في متغيرات مفردة كما يلي:

```
$cars1 = "Volvo";
$cars2 = "BMW";
$cars3 = "Toyota";
```

ولكن؛ ماذا لو كنت ترغب في إنشاء حلقة للسيارات والعثور على واحدة محددة؟ وماذا لو لم يكن لديك 3 سيارات فقط، بل 300؟

الحل هو إنشاء مصفوفة!

يمكن أن تحتوي المصفوفة على العديد من القيم تحت اسم واحد، ويمكنك الوصول إلى القيم من خلال الرجوع إلى رقم الفهرس.

إنشاء مصفوفة

في PHP، يتم استخدام الدالة `array()` لإنشاء مصفوفة:

```
array();
```

في PHP، هناك ثلاثة أنواع من المصفوفات:

- **المصفوفات المفهرسة** – المصفوفات ذات فهرس رقمي.
- **المصفوفات الترابطية** – المصفوفات ذات المفاتيح المسماة.
- **مصفوفات متعددة الأبعاد** – مصفوفات تحتوي على مصفوفة واحدة أو أكثر.

للحصول على طول المصفوفة – نستخدم دالة الـ `count()`

يتم استخدام الدالة `count()` لإرجاع الطول (عدد العناصر) للمصفوفة:

مـــثـــال:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
```

```
echo count($cars);
?>
```

المصفوفات المفهرسة

هناك طريقتان لإنشاء مصفوفات مفهرسة:

يمكن تعيين الفهرس تلقائيًا (يبدأ الفهرس دائمًا من 0)، مثل هذا:

```
$cars = array("Volvo", "BMW", "Toyota");
```

أو يمكن تعيين الفهرس يدويًا:

```
$cars[0] = "Volvo";
$cars[1] = "BMW";
$cars[2] = "Toyota";
```

يقوم المثال التالي بإنشاء مصفوفة مفهرسة باسم `$cars`، ويقوم بتعيين ثلاثة عناصر إليه، ثم يقوم بطباعة نص يحتوي على قيم المصفوفة:

مـــثـــال:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1]
. " and " . $cars[2] . ".";
?>
```

حلقة بمصفوفة مفهرسة

للتنقل بين جميع قيم المصفوفة المفهرسة وطباعتها، يمكنك استخدام حلقة for، مثل هذه:

مـــثـــال:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
$arrlength = count($cars);

for($x = 0; $x < $arrlength; $x++) {
    echo $cars[$x];
    echo "<br>";
}
?>
```

المصفوفات الترابطية

المصفوفات المرتبطة: هي مصفوفات تستخدم مفاتيح معرفة تقوم بتعيينها لها.

هناك طريقتان لإنشاء مصفوفة مرتبطة:

```
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
```

أو:

```
$age['Peter'] = "35";
$age['Ben'] = "37";
$age['Joe'] = "43";
```

يمكن بعد ذلك استخدام المفاتيح المعرفة في البرامج النصية:

مــثـــال:

```
<?php
$age
= array("Peter"=>"35", "Ben"=>"37", "Joe"=>"4
3");
echo "Peter is " . $age['Peter'] . " years
old.";
?>
```

حلقة داخل مصفوفة ترابطية لدخول وطباعة جميع قيم المصفوفة المترابطة، يمكنك استخدام حلقة foreach، مثل هذه:

مـثـــال:

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");

foreach($age as $x => $x_value) {
    echo "Key=" . $x . ", Value=" . $x_value;
    echo "<br>";
}
?>
```

مصفوفات متعددة الأبعاد

في الصفحات السابقة، قمنا بوصف مصفوفات تمثل قائمة واحدة من أزواج مفاتيح/قيم.

ولكن، في بعض الأحيان تريد تخزين القيم باستخدام أكثر من مفتاح. لهذا، لدينا مصفوفات متعددة الأبعاد.

مصفوفة متعدد الأبعاد: عبارة عن مصفوفة تحتوي على مصفوفة واحدة أو أكثر.

تدعم PHP مصفوفات متعددة الأبعاد ذات مستويين أو ثلاثة أو أربعة أو خمسة أو أكثر بعمق. ومع ذلك، يصعب إدارة المصفوفات التي يزيد عمقها عن ثلاثة مستويات لأغلب الأشخاص.

يشير البعد الخاص بالمصفوفة إلى عدد المؤشرات التي تحتاجها لتحديد عنصر.

- ❖ لمصفوفة ثنائية الأبعاد، تحتاج إلى مؤشرين لتحديد عنصر.
- ❖ لمصفوفة ثلاثية الأبعاد، تحتاج إلى ثلاثة مؤشرات لتحديد عنصر.

مصفوفات ثنائية الأبعاد

مصفوفة ثنائية الأبعاد: هي مجموعة من المصفوفات (المصفوفة ثلاثية الأبعاد هي مجموعة من المصفوفات).

أولاً، ألقِ نظرة على الجدول التالي:

الاسم	الكمية	مباعة
Volvo	22	18
BMW	15	13
Saab	5	2
Land Rover	17	15

يمكننا تخزين البيانات من الجدول أعلاه في مصفوفة ثنائية الأبعاد، مثل هذه:

```
$cars = array
  (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
  );
```

تحتوي مصفوفة $cars ثنائية الأبعاد الآن على أربعة مصفوفات، ولها مؤشران: صف وعمود.

للوصول إلى عناصر مصفوفة $cars، يجب أن نشير إلى المؤشرين (الصف والعمود):

مــثـــال:

```
<?php
echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2].".<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2].".<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2].".<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2].".<br>";
?>
```

يمكننا أيضًا وضع حلقة for داخل حلقة for أخرى للحصول على عناصر مصفوفة $cars (لا يزال يتعين علينا الإشارة إلى المؤشرين):

مــثـــال:

```
<?php
for ($row = 0; $row < 4; $row++) {
  echo "<p><b>Row number $row</b></p>";
  echo "<ul>";
  for ($col = 0; $col < 3; $col++) {
    echo "<li>".$cars[$row][$col]."</li>";
  }
  echo "</ul>";
}
?>
```

فرز المصفوفات

يمكنك فرز العناصر في مصفوفة بترتيب أبجدي أو رقمي، تنازليًا أو تصاعديًا.

دوال لفرز المصفوفات

في هذا الفصل، سنتناول دوال فرز المصفوفات التالية:

الدالة	الوظيفة
`sort()`	فرز المصفوفات بترتيب تصاعدي
`rsort()`	فرز المصفوفات بترتيب تنازلي
`asort()`	فرز المصفوفات الترابطية بترتيب تصاعدي، حسب القيمة
`ksort()`	فرز المصفوفات الترابطية بترتيب تصاعدي، حسب المفتاح
`arsort()`	فرز المصفوفات الترابطية بترتيب تنازلي، حسب القيمة
`krsort()`	فرز المصفوفات الترابطية بترتيب تنازلي، حسب المفتاح

فرز المصفوفة بترتيب تصاعدي - sort()

يقوم المثال التالي بفرز عناصر المصفوفة $cars بترتيب أبجدي تصاعديا:

مــثـــال:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
sort($cars);
?>
```

يقوم المثال التالي بفرز عناصر مصفوفة $numbers بترتيب تصاعدي رقمي:

مــثـــال:

```
<?php
$numbers = array(4, 6, 2, 22, 11);
sort($numbers);
?>
```

فرز المصفوفات بترتيب تنازلي – rsort()

المثال التالي يفرز عناصر مصفوفة $cars بالترتيب الأبجدي تنازليا:

مــثـــال:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");
rsort($cars);
?>
```

يقوم المثال التالي بفرز عناصر مصفوفة $numbers بالترتيب الرقمي تنازليا:

مــثـــال:

```
<?php
$numbers = array(4, 6, 2, 22, 11);
rsort($numbers);
?>
```

فرز المصفوفة (ترتيب تصاعدي)، وفقًا للقيمة – asort()

يقوم المثال التالي بفرز مصفوفة مترابطة بترتيب تصاعدي، وفقًا للقيمة:

مــثـــال:

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
asort($age);
?>
```

فرز المصفوفة (ترتيب تصاعدي)، وفقًا للمفتاح – ksort()

المثال التالي يفرز مصفوفة مترابطة بترتيب تصاعدي، وفقًا للمفتاح:

مــثـــال:

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
ksort($age);
?>
```

فرز مصفوفة (ترتيب تنازلي)، وفقا للقيمة – arsort()

مــثـــال:

```
<?php
$age = array("Peter"=>"35", "Ben"=>"37", "Joe"=>"43");
arsort($age);
?>
```

فرز المصفوفة (ترتيب تنازلي)، وفقا للمفتاح – krsort()

المثال التالي يفرز مصفوفة مترابطة بترتيب تنازلي، وفقًا للمفتاح:

مــثـــال:

```
<?php
$age
= array("Peter"=>"35", "Ben"=>"37", "Joe"=>"4
3");
krsort($age);
?>
```

المتغيرات العالمية - Superglobals

تم تقديم Superglobals في PHP 4.1.0، وهي متغيرات مضمنة متوفرة دائمًا في جميع النطاقات.

المتغيرات العالمية - Superglobals

بعض المتغيرات المحددة مسبقًا في PHP هي "superglobals"، مما يعني أنه يمكن الوصول إليها دائمًا، بغض النظر عن النطاق - ويمكنك الوصول إليها من أي دالة أو فئة أو ملف دون الحاجة إلى القيام بأي شيء خاص.

المتغيرات الـ superglobal هي:

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_ENV
- $_COOKIE
- $_SESSION

المتغيرات العالمية الفائقة هي متغيرات مضمنة متوفرة دائمًا في جميع النطاقات.

$GLOBALS

هو متغير عالمي فائق، يستخدم للوصول إلى المتغيرات العالمية من أي مكان في البرنامج النصي لـ PHP (أيضًا من داخل الدوال أو الطرق).

يخزن PHP جميع المتغيرات العامة في مصفوفة تسمى **$GLOBALS[*index*]**.

index يحمل اسم المتغير.

يوضح المثال التالي كيفية استخدام المتغير العالمي $GLOBALS:

مــثـــال:

```
<?php
$x = 75;
$y = 25;

function addition() {
    $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}

addition();
echo $z;
?>
```

$_SERVER

عبارة عن متغير super global يحتوي على معلومات حول الرؤوس والمسارات ومواقع البرامج النصية.

يوضح المثال التالي كيفية استخدام بعض العناصر في $_SERVER:

مـــثـــال:

```
<?php
echo $_SERVER['PHP_SELF'];
echo "<br>";
echo $_SERVER['SERVER_NAME'];
echo "<br>";
echo $_SERVER['HTTP_HOST'];
echo "<br>";
echo $_SERVER['HTTP_REFERER'];
echo "<br>";
echo $_SERVER['HTTP_USER_AGENT'];
echo "<br>";
echo $_SERVER['SCRIPT_NAME'];
?>
```

يسرد الجدول التالي أهم العناصر التي يمكن إدخالها داخل $_SERVER:

إرجاع اسم ملف البرنامج النصي الذي يتم تنفيذه حاليًا:

```
1.    $_SERVER['PHP_SELF']
```

إرجاع إصدار واجهة البوابة العامة (CGI) "Common Gateway Interface" التي يستخدمها الخادم:

```
2.    $_SERVER['GATEWAY_INTERFACE']
```

إرجاع عنوان IP الخاص بالخادم المضيف:

```
3.    $_SERVER['SERVER_ADDR']
```

إرجاع اسم الخادم المضيف (مثل www.w3schools.com):

```
4.    $_SERVER['SERVER_NAME']
```

إرجاع سلسلة تعريف الخادم (مثل Apache/2.2.24):

```
5.    $_SERVER['SERVER_SOFTWARE']
```

إرجاع اسم ومراجعة بروتوكول المعلومات (مثل HTTP / 1.1):

```
6.    $_SERVER['SERVER_PROTOCOL']
```

إرجاع طريقة الطلب المستخدمة للوصول إلى الصفحة (مثل POST):

```
7.    $_SERVER['REQUEST_METHOD']
```

إرجاع الطابع الزمني لبداية الطلب (مثل 1377687496):

```
8.    $_SERVER['REQUEST_TIME']
```

إرجاع سلسلة الاستعلام إذا تم الوصول إلى الصفحة عبر سلسلة استعلام:

```
9.    $_SERVER['QUERY_STRING']
```

إرجاع رأس قبول من الطلب الحالي:

```
10.    $_SERVER['HTTP_ACCEPT']
```

إرجاع رأس Accept_Charset من الطلب الحالي (مثل UTF-8 و ISO-8859-1):

```
11.    $_SERVER['HTTP_ACCEPT_CHARSET']
```

إرجاع رأس المضيف من الطلب الحالي:

```
12.    $_SERVER['HTTP_HOST']
```

إرجاع عنوان URL الكامل للصفحة الحالية (غير موثوق لأنه لا يدعمها جميع وكلاء المستخدم):

```
13.    $_SERVER['HTTP_REFERER']
```

يتم الاستعلام عن البرنامج النصي من خلال بروتوكول HTTP الآمن:

```
14.    $_SERVER['HTTPS']
```

إرجاع عنوان IP من حيث يعرض المستخدم الصفحة الحالية:

```
15.    $_SERVER['REMOTE_ADDR']
```

إرجاع اسم المضيف من حيث يعرض المستخدم الصفحة الحالية:

```
16.    $_SERVER['REMOTE_HOST']
```

إرجاع المنفذ المستخدم على جهاز المستخدم للتواصل مع خادم الويب:

```
17.    $_SERVER['REMOTE_PORT']
```

إرجاع اسم المسار المطلق للبرنامج النصي الحالي:

```
18.    $_SERVER['SCRIPT_FILENAME']
```

تُرجع القيمة المعطاة إلى توجيه SERVER_ADMIN في ملف تكوين خادم الويب (إذا كان البرنامج النصي يعمل على مضيف افتراضي، فستكون القيمة المحددة لهذا المضيف الافتراضي) (مثل someone@w3schools.com):

```
19.    $_SERVER['SERVER_ADMIN']
```

إرجاع المنفذ على جهاز الخادم الذي يستخدمه خادم الويب للاتصال (مثل 80):

```
20.    $_SERVER['SERVER_PORT']
```

إرجاع إصدار الخادم واسم المضيف الافتراضي الذي تمت إضافته إلى الصفحات التي أنشأها الخادم:

```
21.    $_SERVER['SERVER_SIGNATURE']
```

إرجاع المسار المستند لنظام الملفات إلى البرنامج النصي الحالي:

```
22.    $_SERVER['PATH_TRANSLATED']
```

إرجاع مسار البرنامج النصي الحالي:

```
23.    $_SERVER['SCRIPT_NAME']
```

إرجاع عنوان URI للصفحة الحالية:

```
24.    $_SERVER['SCRIPT_URI']
```

$_REQUEST

يستخدم لجمع البيانات بعد تقديم نموذج HTML.

يوضح المثال التالي نموذجًا يحتوي على حقل إدخال وزر إرسال. عندما يرسل المستخدم البيانات بالنقر فوق "submit"، يتم إرسال بيانات النموذج إلى الملف المحدد في خاصية الإجراء "action" للوسم <form>. في هذا المثال، نشير إلى هذا الملف نفسه لمعالجة بيانات النموذج. إذا كنت ترغب في استخدام ملف PHP آخر لمعالجة بيانات النموذج، فاستبدل ذلك باسم الملف الذي تختاره. بعد ذلك، يمكننا استخدام $_REQUEST لجمع قيم حقل الإدخال:

مـثــال:

```
<html>
<body>
<form method="post" action="<?php echo $_SERV
ER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
  <input type="submit">
</form>
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // collect value of input field
    $name = $_REQUEST['fname'];
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name;
    }
}
?>
</body>
</html>
```

$_POST

يُستخدم لجمع بيانات النموذج بعد إرسال نموذج HTML من خلال "method = "post. يستخدم $_POST أيضًا على نطاق عالمي لتمرير المتغيرات.

يوضح المثال التالي نموذجًا يحتوي على حقل إدخال وزر إرسال. عندما يرسل المستخدم البيانات بالنقر فوق "إرسال"، يتم إرسال بيانات النموذج إلى الملف المحدد في خاصية الإجراء للوسم <form>. في هذا المثال، نشير إلى الملف نفسه لمعالجة بيانات النموذج. إذا كنت ترغب في استخدام ملف PHP آخر لمعالجة بيانات النموذج، فاستبدل ذلك باسم الملف الذي تختاره. بعد ذلك، يمكننا استخدام $_POST لجمع قيمة حقل الإدخال:

مــثـــال:

```
<html>
<body>
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
  Name: <input type="text" name="fname">
        <input type="submit">
</form>
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST") {
    // جمع القيم من حقول الإدخال
    $name = $_POST['fname'];
    if (empty($name)) {
        echo "Name is empty";
    } else {
        echo $name;
    }
}
?>
</body>
</html>
```

$_GET

تستخدم لجمع بيانات النموذج بعد إرسال نموذج HTML بـ "method = "get.

يمكنه أيضًا جمع البيانات المرسلة في عنوان URL.

افترض أن لدينا صفحة HTML تحتوي على ارتباط تشعبي مع معلمات:

```
<html>
<body>

<a href="test_get.php?subject=PHP&web=W3schools.com">Test $GET</a>

</body>
</html>
```

عندما ينقر المستخدم على الرابط "Test GET"، يتم إرسال المعلمتين "topic" و "web" إلى "test_get.php"، ثم يمكنك الوصول إلى قيمهما في "test_get.php" بـ $_GET.

المثال التالي يوضح الكود الموجود في "test_get.php":

مــثـــال:

```
<?php
echo "Study " . $_GET['subject'] . " at
" . $_GET['web'];
?>
```

نماذج php

معالجة نماذج php

تستخدم الـ $_GET و $_POST لجمع بيانات النموذج.

نموذج html بسيط

المثال التالي يعرض نموذج html بسيط بحقلي إدخال وزر إرسال:

مــثـــال:

```
<html>
<body>

<form action="welcome.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

عندما يملأ المستخدم النموذج أعلاه وينقر فوق زر إرسال، يتم إرسال بيانات النموذج للمعالجة إلى ملف PHP باسم "welcome.php". يتم إرسال بيانات النموذج باستخدام طريقة HTTP POST.

لعرض البيانات المرسلة، يمكنك ببساطة طباعة جميع المتغيرات. تبدو "welcome.php" كالتالي:

```
<html>
<body>

Welcome <?php echo $_POST["name"]; ?><br>
Your email address is: <?php echo $_POST["email"]; ?>

</body>
</html>
```

يجب أن يكون الإخراج يشبه هذا:

```
Welcome John
Your email address is john.doe@example.com
```

يمكن تحقيق نفس النتيجة باستخدام طريقة HTTP GET:

مــثـــال:

```
<html>
<body>

<form action="welcome_get.php" method="get">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>

</body>
</html>
```

و "welcome_get.php" تبدو كالتالي:

```
<html>
<body>

Welcome <?php echo $_GET["name"]; ?><br>
Your email address is: <?php echo $_GET["email"];
?>
```

```
</body>
</html>
```

الكود السابق بسيط للغاية. ومع ذلك، فإن أهم شيء مفقود. تحتاج إلى التحقق من صحة بيانات النموذج لحماية البرنامج النصي من التعليمات البرمجية الضارة.

فكر في الأمان عند معالجة نماذج PHP!

لا تحتوي هذه الصفحة على أي تحقق من صحة النموذج، ولكنها توضح فقط كيف يمكنك إرسال واسترداد بيانات النموذج.

ومع ذلك، ستظهر الصفحات التالية كيفية معالجة نماذج PHP مع مراعاة الأمان! التحقق الصحيح من بيانات النموذج مهم لحماية النموذج الخاص بك من المتسللين ومرسلي البريد العشوائي!

GET مقابل POST

ينشئ كل من GET و POST مصفوفة (مثال المصفوفة (key1 => value1، key2 => value2، key3 => value3 ، ...)). تحتوي هذه المصفوفة على أزواج مفاتيح/قيم، حيث تكون المفاتيح هي أسماء عناصر تحكم النموذج والقيم هي بيانات الإدخال من المستخدم.

كل من GET و POST يعاملان كـ $_GET و $_POST. هذه هي super globals، مما يعني أنه يمكن الوصول إليها دائمًا، بغض النظر عن النطاق – ويمكنك الوصول إليها من أي دالة أو فئة أو ملف دون الحاجة للقيام بأي شيء خاص.

$_GET : عبارة عن مجموعة من المتغيرات التي تم تمريرها إلى البرنامج النصي الحالي عبر معلمات URL.

$_POST : هو مجموعة من المتغيرات التي تم تمريرها إلى البرنامج النصي الحالي بطريقة HTTP POST.

متى نستخدم GET؟

تكون المعلومات المرسلة من نموذج باستخدام طريقة GET مرئية للجميع (يتم عرض جميع الأسماء والقيم المتغيرة في عنوان URL). لدى GET أيضًا حدود على كمية المعلومات المراد إرسالها. القيد هو حوالي 2000 حرف. ومع ذلك، نظرًا لعرض المتغيرات في عنوان URL، فمن الممكن وضع إشارة مرجعية على الصفحة. هذا يمكن أن يكون مفيدا في بعض الحالات.

يمكن استخدام GET لإرسال البيانات غير الحساسة.

ملاحظة: يجب ألا تستخدم GET مطلقًا لإرسال كلمات المرور أو غيرها من المعلومات الحساسة!

متى نستخدم POST

المعلومات المرسلة من نموذج بطريقة POST **غير مرئية للآخرين** (جميع الأسماء/القيم مضمنة في نص طلب HTTP) **وليس لها حدود** على كمية المعلومات المراد إرسالها.

علاوة على ذلك، تدعم POST الوظائف المتقدمة مثل دعم الإدخال الثنائي متعدد الأجزاء أثناء تحميل الملفات للخادم.

ولكن، نظرًا لعدم عرض المتغيرات في عنوان URL، لا يمكن وضع إشارة مرجعية على الصفحة.

> يفضل المطورون تطبيق POST لإرسال بيانات النموذج.

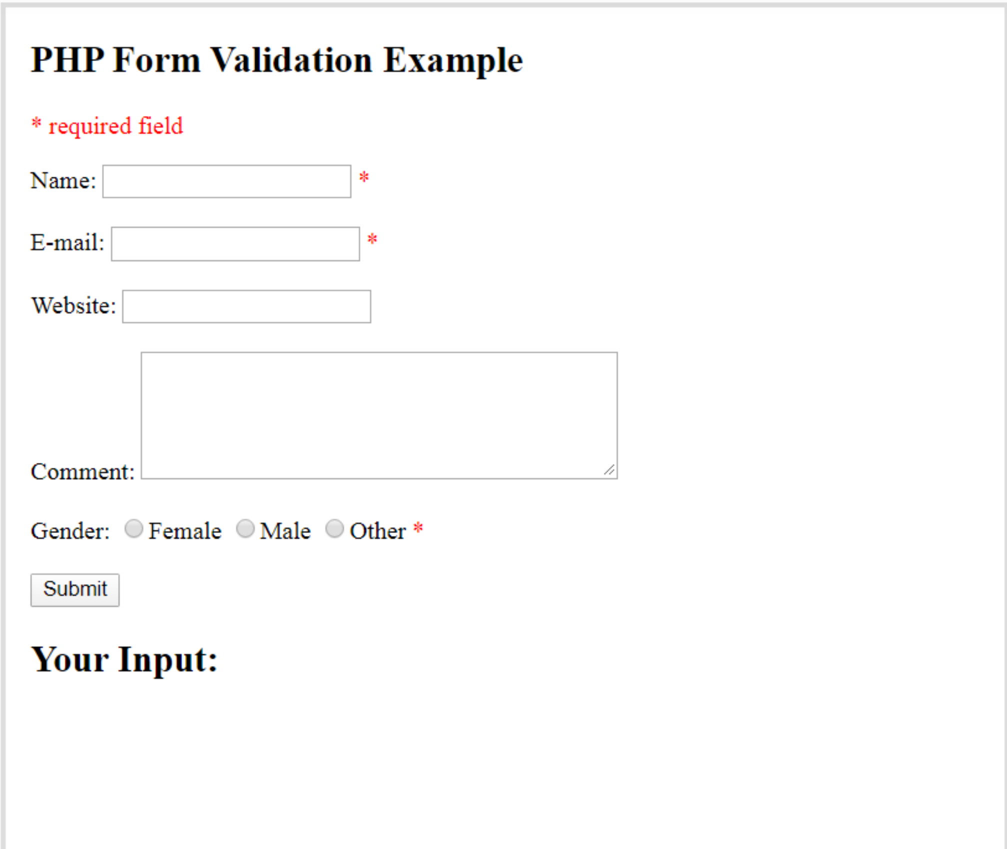
التالي، سنرى كيف يمكننا معالجة نماذج php بطريقة آمنة.

التحقق من صحة نموذج PHP

في هذا الفصل والفصول التالية سنرى كيف نستخدم php للتحقق من صحة بيانات النموذج.

فكر في الأمان عند معالجة نماذج PHP!
ستوضح هذه الصفحات كيفية معالجة نماذج PHP مع مراعاة الأمان. التحقق الصحيح من بيانات النموذج مهم لحماية النموذج الخاص بك من المتسللين ومرسلي البريد العشوائي!

يحتوي نموذج HTML الذي سنعمل عليه في هذه الفصول، على حقول الإدخال المختلفة: حقول النص المطلوبة والاختيارية وأزرار الاختيار وزر الإرسال:



قواعد التحقق من صحة النموذج أعلاه هي كما يلي:

قواعد التحقق	الحقل
مطلوب. + يجب أن يحتوي فقط على حروف ومساحات بيضاء	الاسم
مطلوب. + يجب أن يحتوي على عنوان بريد إلكتروني صالح (بـ @ و.)	الإيميل
اختياري. إذا كان موجودًا، فيجب أن يحتوي على عنوان URL صالح	الموقع الإلكتروني
اختياري. حقل إدخال متعدد الأسطر (textarea)	التعليق
اختياري. يجب اختيار واحد فقط	الجنس

أولا سنلقي نظرة على خريطة نموذج كود html.

الحقول النصية

حقول الاسم، الإيميل، والموقع الإلكتروني هي عناصر ادخال النصوص، وحقل التعليق textarea.

كود الـ html يشبه هذا:

```
Name: <input type="text" name="name">
E-mail: <input type="text" name="email">
Website: <input type="text" name="website">
Comment: <textarea name="comment" rows="5" cols="40"></t
extarea>
```

الأزرار الاختيارية (Radio Buttons)

حقول الجنس عبارة عن أزرار اختيارية وكود html يبدو كالتالي:

```
Gender:
<input type="radio" name="gender" value="female">
Female
<input type="radio" name="gender" value="male">
Male
<input type="radio" name="gender" value="other">
Other
```

عنصر النموذج

كود الـ html للنموذج يبدو كالتالي:

```
<form method="post" action="<?php echo htmlspecia
lchars($_SERVER["PHP_SELF"]);?>">
```

عند إرسال النموذج، يتم إرسال بيانات النموذج باستخدام "method="post.

ما هو المتغير [$_SERVER["PHP_SELF"] ؟

هو متغير عالمي فائق يُرجع اسم ملف البرنامج النصي الذي يتم تنفيذه حاليًا.

وهو يرسل بيانات النموذج المرسلة إلى الصفحة نفسها، بدلاً من القفز إلى صفحة مختلفة. بهذه الطريقة، سيحصل المستخدم على رسائل خطأ في نفس الصفحة كنموذج.

ما هي دالة الـ `htmlspecialchars()` **؟**

تحول الحروف الخاصة إلى كيانات HTML. هذا يعني أنه سيحل محل حروف HTML مثل < و > بـ < و >. هذا يمنع المهاجمين من استغلال الأكواد عن طريق حقن أكواد HTML أو Javascript (هجمات البرمجة النصية عبر المواقع) في النماذج.

ملاحظة مهمة جدا في حماية نماذج php

المتغير ["PHP_SELF"]$_SERVER يمكن أن يستخدمه المخترقين!

إذا PHP_SELF تستخدم في صفحتك، عندها يمكن للمستخدم ادخال سلاش / ثم تنفيذ بعض أوامر البرمجة النصية.

ما هي دالة الـ htmlspecialchars() ؟

البرمجة النصية لعبور المواقع (XSS) هي نوع من الثغرات الأمنية في أمن الحاسوب الموجودة عادة في تطبيقات الويب. يمكّن XSS المهاجمين من حقن البرنامج النصي من جانب العميل في صفحات الويب التي يشاهدها المستخدمون الآخرون.

افترض أن لدينا النموذج التالي في صفحة باسم"test_form.php" :

```
<form method="post" action="<?php echo
$_SERVER["PHP_SELF"];?>">
```

```

```

الآن، إذا قام المستخدم بإدخال عنوان URL العادي في شريط العنوان مثل "http://www.example.com/test_form.php"، فسيتم ترجمة الكود السابق لـ :

```
<form method="post" action="test_form.php">
```

حتى الان جيد جدا.

ومع ذلك، ضع في اعتبارك أن المستخدم يدخل عنوان URL التالي في شريط العناوين:

```
http://www.example.com/test_form.php/%22%3E%3Cscript%3Ealert('hacked')%3C/script%3E
```

في هذه الحالة، سيتم ترجمة الكود السابق لـ:

```
<form method="post" action="test_form.php/"><script>alert('hacked')</script>
```

يضيف هذا الكود وسم `script` وأمر `alert`. وعندما يتم تحميل الصفحة، سيتم تنفيذ كود JavaScript (سيرى المستخدم مربع التنبيه). هذا مثال بسيط وغير ضار على كيفية استغلال المتغير `PHP_SELF`.

كن على علم بأنه يمكن إضافة أي أكواد JavaScript داخل وسم `<script>`! يمكن للمتسلل إعادة توجيه المستخدم إلى ملف على خادم آخر، ويمكن لهذا الملف أن يحتفظ بكود خبيث يمكنه تغيير المتغيرات العامة أو إرسال النموذج إلى عنوان آخر لحفظ بيانات المستخدم، على سبيل المثال.

كيف تتجنب استغلالات ["PHP_SELF"]$_SERVER؟

يمكن تجنبهن باستخدام دالة `htmlspecialchars()`.

كود النموذج يجب أن يكون كالتالي:

```
<form method="post" action="<?php echo htmlspecialchars($_SERVER["PHP_SELF"]);?>">
```

الدالة `htmlspecialchars()` تغير الحروف الخاصة لكيانات **html**. الآن إذا حاول المستخدم استغلال المتغير `PHP_SELF`، فسيؤدي ذلك إلى الإخراج التالي:

```
<form method="post" action="test_form.php/&quot;&gt;&lt;script&gt;alert('hacked')&lt;/script&gt;">
```

فشلت محاولة الإستغلال، ولم يحدث أي ضرر!

التحقق من صحة بيانات النموذج بستخدام php

أول شيء سنفعله هو تمرير جميع المتغيرات من خلال دالة ()htmlspecialchars في PHP.

عندما نستخدم دالة ()htmlspecialchars؛ إذا حاول المستخدم إرسال ما يلي في حقل النص:

```
<script>location.href('http://www.hacked.com')</script>
```

لن يتم تنفيذ هذا؛ لأنه سيتم حفظه ككود html، مثل هذا:

```
&lt;script&gt;location.href('http://www.hacked.com')&lt;/script&gt;
```

الكود الآن آمن ليتم عرضه على صفحة أو داخل بريد إلكتروني.

سنفعل شيئين آخرين عندما يقوم المستخدم بإرسال النموذج:

١- إزالة الأحرف غير الضرورية (مساحة إضافية، علامة تبويب، سطر جديد) من بيانات إدخال المستخدم (بدالة ()trim)

٢- إزالة الخطوط المائلة العكسية (\) من بيانات إدخال المستخدم (بدالة ()stripslashes)

والخطوة التالية هي إنشاء دالة ستقوم بكل عمليات التدقيق لنا (والتي تعد أكثر ملاءمة من كتابة الكود نفسه مرارًا وتكرارًا).

سنقوم بتسمية الدالة `test_input()`.

الآن، يمكننا التحقق من كل متغير `$_POST` بدالة `test_input()`، والنص البرمجي يبدو كما يلي:

مــثـــال:

```
<?php
// تعريف المتغيرات ووضع فيها قيم فارغة
$name = $email = $gender = $comment =
$website = "";
```

```
if ($_SERVER["REQUEST_METHOD"] == "POST") {
  $name = test_input($_POST["name"]);
  $email = test_input($_POST["email"]);
  $website = test_input($_POST["website"]);
  $comment = test_input($_POST["comment"]);
  $gender = test_input($_POST["gender"]);
}

function test_input($data) {
  $data = trim($data);
  $data = stripslashes($data);
  $data = htmlspecialchars($data);
  return $data;
}
?>
```

لاحظ أنه في بداية البرنامج النصي، نتحقق من إرسال النموذج باستخدام $_SERVER["REQUEST_METHOD"]. إذا كان REQUEST_METHOD هو POST، فسيتم إرسال النموذج – ويجب التحقق من صحته. إذا لم يتم تقديمها، تخطي التحقق من الصحة وعرض نموذج فارغ.

ومع ذلك، في المثال السابق، جميع حقول الإدخال اختيارية. البرنامج النصي يعمل بشكل جيد حتى لو لم يقم المستخدم بإدخال أي بيانات.

الخطوة التالية هي جعل حقول الإدخال مطلوبة وإنشاء رسائل خطأ إذا لزم الأمر.

طلب الحقول

يوضح هذا الفصل كيفية جعل حقول الإدخال مطلوبة وإنشاء رسائل خطأ إذا لزم الأمر.

من جدول قواعد التحقق من الصحة في الصفحة السابقة، نرى أن حقول "الاسم" و "البريد الإلكتروني" و "الجنس" مطلوبة. لا يمكن أن تكون هذه الحقول فارغة ويجب ملؤها في نموذج HTML.

الحقل	قواعد التحقق
الاسم	مطلوب. + يجب أن يحتوي فقط على حروف ومساحات بيضاء
الإيميل	مطلوب. + يجب أن يحتوي على عنوان بريد إلكتروني صالح (بـ @ و.)
الموقع الإلكتروني	اختياري. إذا كان موجودًا، فيجب أن يحتوي على عنوان URL صالح
التعليق	اختياري. حقل إدخال متعدد الأسطر (textarea)
الجنس	اختياري. يجب اختيار واحد فقط

في الفصل السابق، كانت جميع حقول الإدخال اختيارية.

في التعليمات البرمجية التالية أضفنا بعض المتغيرات الجديدة:

$nameErr

$emailErr

$genderErr

$websiteErr

سوف تحتفظ متغيرات الخطأ هذه برسائل خطأ للحقول المطلوبة. لقد أضفنا أيضًا جملة else if لكل متغير $_POST. هذا يتحقق إذا كان المتغير $_POST فارغًا (بالدالة empty()). إذا كانت فارغة، يتم تخزين رسالة الخطأ في متغيرات الخطأ المختلفة، وإذا لم تكن فارغة، فإنها ترسل بيانات إدخال المستخدم من خلال دالة test_input():

مـــثـــال:

```
<?php
// تـعـريـف الـمـتـغـيـرات ووضع قـيـم فـارغـة
$nameErr = $emailErr = $genderErr =
$websiteErr = "";
$name = $email = $gender = $comment =
$website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
  }
```

```
  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
  } else {
    $email = test_input($_POST["email"]);
  }

  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website =
test_input($_POST["website"]);
  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment =
test_input($_POST["comment"]);
  }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
    $gender = test_input($_POST["gender"]);
  }
}
?>
```

عرض رسائل الأخطاء

بعد ذلك في نموذج HTML، نضيف نصًا صغيرًا بعد كل حقل مطلوب، والذي ينشئ رسالة الخطأ الصحيحة إذا لزم الأمر (أي إذا حاول المستخدم إرسال النموذج دون ملء الحقول المطلوبة):

مـثــال:

```
<form method="post" action="<?php echo htmlsp
ecialchars($_SERVER["PHP_SELF"]);?>">

Name: <input type="text" name="name">
<span class="error">* <?php echo $nameErr;?><
/span>
<br><br>
E-mail:
<input type="text" name="email">
<span class="error">* <?php echo $emailErr;?>
</span>
<br><br>
Website:
<input type="text" name="website">
<span class="error"><?php echo $websiteErr;?>
</span>
<br><br>
Comment: <textarea name="comment" rows="5" co
ls="40"></textarea>
<br><br>
Gender:
<input type="radio" name="gender" value="fema
le">Female
<input type="radio" name="gender" value="male
">Male
```

```
<input type="radio" name="gender" value="othe
r">Other
<span class="error">* <?php echo $genderErr;?
></span>
<br><br>
<input type="submit" name="submit" value="Sub
mit">

</form>
```

والخطوة التالية هي التحقق من صحة بيانات الإدخال، وهي "هل يحتوي حقل الاسم فقط على أحرف ومسافة بيضاء؟"، و "هل يحتوي حقل البريد الإلكتروني على بنية جملة صالح لعنوان البريد الإلكتروني؟"، وإذا تم ملؤه، " هل يحتوي حقل موقع الويب على عنوان URL صالح؟ ".

تحقق من صحة البريد الإلكتروني وعنوان URL

يعرض الكود التالي طريقة بسيطة للتحقق مما إذا كان حقل الاسم يحتوي فقط على أحرف ومسافة بيضاء. إذا كانت قيمة حقل الاسم غير صالحة، فقم بتخزين رسالة خطأ:

```
$name = test_input($_POST["name"]);
if (!preg_match("/^[a-zA-Z ]*$/",$name)) {
  $nameErr = "Only letters and white space allowed";
}
```

تقوم الدالة `preg_match()` بالبحث في سلسلة عن نموذج (pattern)، وإرجاع صحيح إذا كان النموذج موجودًا، وغير صحيحة لغير ذلك.

التحقق من صحة البريد الإلكتروني

الطريقة الأسهل والأكثر أمانًا للتحقق مما إذا كان عنوان البريد الإلكتروني جيد التنسيق هو استخدام دالة `filter_var()` من PHP.

في الكود التالي، إذا لم يكن عنوان البريد الإلكتروني جيدًا، فعليك تخزين رسالة خطأ:

```
$email = test_input($_POST["email"]);
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
  $emailErr = "Invalid email format";
}
```

التحقق من صحة URL

يعرض الرمز أدناه طريقة للتحقق من صحة بناء جملة عنوان URL (يسمح هذا التعبير المنتظم أيضًا الشرط في عنوان URL). إذا كان بناء جملة عنوان URL غير صالح، فقم بتخزين رسالة خطأ:

```
$website = test_input($_POST["website"]);
if
(!preg_match("/\b(?:(?:https?|ftp):\/\/|www\.)[-
a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-
9+&@#\/%=~_|]/i",$website)) {
  $websiteErr = "Invalid URL";
}
```

التحقق من صحة الإيميل

الأن، البرنامج النصي يبدوا مثل:

مــثـــال:

```
<?php
// define variables and set to empty values
$nameErr = $emailErr = $genderErr =
$websiteErr = "";
$name = $email = $gender = $comment =
$website = "";

if ($_SERVER["REQUEST_METHOD"] == "POST") {
  if (empty($_POST["name"])) {
    $nameErr = "Name is required";
  } else {
    $name = test_input($_POST["name"]);
    // check if name only contains letters
and whitespace
    if (!preg_match("/^[a-zA-Z ]*$/",$name))
{
      $nameErr = "Only letters and white
space allowed";
    }
  }

  if (empty($_POST["email"])) {
    $emailErr = "Email is required";
```

```
  } else {
    $email = test_input($_POST["email"]);
    // check if e-mail address is well-formed
    if (!filter_var($email,
FILTER_VALIDATE_EMAIL)) {
      $emailErr = "Invalid email format";
    }
  }

  if (empty($_POST["website"])) {
    $website = "";
  } else {
    $website = test_input($_POST["website"]);
    // check if URL address syntax is valid
(this regular expression also allows dashes
in the URL)
    if (!preg_match("/\b(?:(?:https?|ftp):\/\
/|www\.)[-a-z0-9+&@#\/%?=~_|!:,.;]*[-a-z0-
9+&@#\/%=~_|]/i",$website)) {
      $websiteErr = "Invalid URL";
    }
  }

  if (empty($_POST["comment"])) {
    $comment = "";
  } else {
    $comment = test_input($_POST["comment"]);
  }

  if (empty($_POST["gender"])) {
    $genderErr = "Gender is required";
  } else {
```

```
    $gender = test_input($_POST["gender"]);
  }
}
?>
```

الخطوة التالية: هي إظهار كيفية منع النموذج من إفراغ جميع حقول الإدخال عندما يقوم المستخدم بإرسال النموذج.

لإظهار القيم في حقول الإدخال بعد قيام المستخدم بالضغط على زر الإرسال، نضيف القليل من نص PHP داخل خاصية القيمة لحقول الإدخال التالية: الاسم، البريد الإلكتروني، وموقع الويب. في حقل textarea الخاص بالتعليق، نضع النص بين علامتي <textarea> و </textarea>. يقوم البرنامج النصي الصغير بإخراج قيمة متغيرات $name، $email و $web و $comment.

بعد ذلك، نحتاج أيضًا إلى إظهار زر الاختيار الذي تم فحصه. لهذا، يجب علينا معالجة الخاصية المحددة (وليس خاصية القيمة لأزرار الاختيار):

مــثــال:

```
Name: <input type="text" name="name" value="<
?php echo $name;?>">

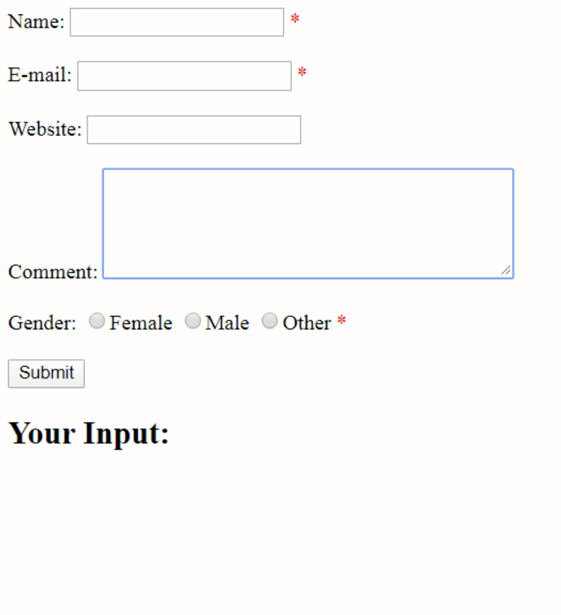
E-
mail: <input type="text" name="email" value="
<?php echo $email;?>">
Website: <input type="text" name="website" va
lue="<?php echo $website;?>">
Comment: <textarea name="comment" rows="5" co
```

```
ls="40"><?php echo $comment;?></textarea>
Gender:
<input type="radio" name="gender"
<?php if (isset($gender) &&
$gender=="female") echo "checked";?>
value="female">Female
<input type="radio" name="gender"
<?php if (isset($gender) &&
$gender=="male") echo "checked";?>
value="male">Male
<input type="radio" name="gender"
<?php if (isset($gender) &&
$gender=="other") echo "checked";?>
value="other">Other
```

مثال كامل على النموذج

فيما يلي الرمز الكامل لمثال التحقق من صحة نموذج PHP:

مثال:



php متقدمة

التاريخ والوقت

يتم استخدام دالة `date()` لتنسيق التاريخ و / أو الوقت.

دالة `date()`

بنية الجملة

```
date(format,timestamp)
```

المعلمة	الوصف
format	مطلوب. يحدد تنسيق الطابع الزمني
timestamp	اختياري. يحدد الطابع الزمني. الافتراضي هو التاريخ والوقت الحالي

الطابع الزمني هو سلسلة من الأحرف، تدل على التاريخ و / أو الوقت الذي حدث فيه حدث معين.

جلب التاريخ

تحدد معلمة التنسيق المطلوبة للدالة `date()` كيفية تنسيق التاريخ (أو الوقت).

فيما يلي بعض الأحرف شائعة الاستخدام للتواريخ:

- ❖ d – يمثل اليوم (من 01 إلى 31)
- ❖ m – يمثل الشهر (01 إلى 12)
- ❖ Y – يمثل السنة (بأربعة أرقام)
- ❖ l (صغيرة 'L') – يمثل يوم الأسبوع

يمكن أيضًا إدراج أحرف أخرى، مثل "/" أو "." أو "–" بين الأحرف لإضافة تنسيق إضافي.

المثال التالي ينسق تاريخ اليوم بثلاث طرق مختلفة:

مــثـــال:

```
<?php
echo "Today is " . date("Y/m/d") . "<br>";
echo "Today is " . date("Y.m.d") . "<br>";
echo "Today is " . date("Y-m-d") . "<br>";
echo "Today is " . date("l");
?>
```

ملاحظة – سنة حقوق النسخ تلقائية

استخدم دالة `date()` لتحديث سنة حقوق النسخ تلقائيا على موقعك:

مــثـــال:

```
&copy; 2010-<?php echo date("Y");?>
```

جلب الوقت

فيما يلي بعض الأحرف التي يتم استخدامها بشكل شائع للأوقات:

- ❖ H – تنسيق 24 ساعة (00 إلى 23)
- ❖ h – تنسيق 12 ساعة (01 إلى 12)
- ❖ i – دقائق (00 إلى 59)
- ❖ s – ثواني (00 إلى 59)
- ❖ a – الأحرف الصغيرة (am أو pm)

المثال التالي يخرج الوقت الحالي بالتنسيق المحدد:

مــثـــال:

```
<?php
echo "The time is " . date("h:i:sa");
?>
```

لاحظ أن دالة `date()` ستعيد التاريخ / الوقت الحالي للخادم!

جلب منطقتك الزمنية

إذا كان الوقت المرجع غير صحيح، فربما يرجع ذلك إلى أن الخادم الخاص بك موجود في بلد آخر أو تم إعداده لمنطقة زمنية مختلفة.

لذلك، إذا كنت بحاجة إلى أن يكون الوقت صحيحًا وفقًا لموقع معين، يمكنك تعيين المنطقة الزمنية التي تريد استخدامها.

المثال التالي يعين المنطقة الزمنية إلى "America/New_York"، ثم يخرج الوقت الحالي بالتنسيق المحدد:

مـــثـــال:

```
<?php
date_default_timezone_set("America/New_York");
echo "The time is " . date("h:i:sa");
?>
```

إنشاء تارخ بدالة mktime()

المعلمة *timestamp* الإختيارية في دالة date() تعرف الطابع الزمني.

الدالة ()mktime ترجع الطابع الزمني في نظام يونكس للتاريخ.

يحتوي الطابع الزمني الخاص بـ Unix على عدد الثواني بين فترة (January 1 1970 00:00:00 GMT) والوقت المحدد.

بينة الجملة

```
mktime(hour, minute, second, month, day, year)
```

المثال التالي سينشئ تاريخ ووقت بدالة الـ ()date من أرقام مدخلة للمعلمات في دالة ()mktime:

مــثـــال:

```
<?php
$d=mktime(11, 14, 54, 8, 12, 2014);
echo "Created date is " . date("Y-m-d
h:i:sa", $d);
?>
```

انشاء تاريخ من سلسلة بدالة ()strtotime

دالة الـ ()strtotime تستخدم لتحويل تاريخ نصي قابل للقراءة البشرية للطابع الزمني لـ unix (عدد الثواني منذ ١ يناير ١٩٧٠ ٠٠:٠٠:٠٠ بتوقيت جرينتش).

بنية الجملة

```
strtotime(time, now)
```

المثال التالي سينشئ تاريخ ووقت من دالة ()strtotime:

مـثـــال:

```
<?php
$d=strtotime("10:30pm April 15 2014");
echo "Created date is " . date("Y-m-d
h:i:sa", $d);
?>
```

php ذكية جدا في تحويل سلسلة لتاريخ، حتى تتمكن من وضع قيم مختلفة:

مـثـــال:

```
<?php
$d=strtotime("tomorrow");
echo date("Y-m-d h:i:sa", $d) . "<br>";

$d=strtotime("next Saturday");
echo date("Y-m-d h:i:sa", $d) . "<br>";

$d=strtotime("+3 Months");
echo date("Y-m-d h:i:sa", $d) . "<br>";
?>
```

على كل حال، دالة ()strtotime ليست مثالية، لذا تذكر التحقق من السلاسل التي تضعها هناك.

أمثلة أكثر على التاريخ:

المثال التالي سيخرج تواريخ أيام السبت الست القادمة:

مــثـــال:

```
<?php
$startdate = strtotime("Saturday");
$enddate = strtotime("+6 weeks",
$startdate);

while ($startdate < $enddate) {
  echo date("M d", $startdate) . "<br>";
  $startdate = strtotime("+1 week",
$startdate);
}
?>
```

المثال التالي سيخرج عدد الأيام من الان إلى الرابع من يوليو:

مــثـــال:

```
<?php
$d1=strtotime("July 04");
$d2=ceil(($d1-time())/60/60/24);
echo "There are " . $d2 ." days until 4th of
July.";
?>
```

تضمين الملفات في php

تأخذ جملة include (أو require) كل النصوص/الأكواد/الترميزات الموجود في الملف المحدد ونسخه في الملف الذي يستخدم جملة التضمين.

يعد تضمين الملفات مفيدًا للغاية عندما تريد تضمين نفس PHP أو HTML أو نص في صفحات متعددة لموقع ويب.

جملتي include و require

من الممكن إدراج محتوى ملف PHP واحد في ملف PHP آخر (قبل أن ينفذه الخادم)، بجملتي include و require.

جملتي الطلب التضمين "include" و الطلب "require" متطابقتين، إلا عند الفشل:

- **require:** يؤدي إلى حدوث خطأ فادح (E_COMPILE_ERROR) وإيقاف البرنامج النصي.
- **Include:** سيصدر تحذير فقط (E_WARNING) وسيستمر البرنامج النصي.

لذلك، إذا كنت تريد أن يستمر التنفيذ وتُظهر للمستخدمين الإخراج، حتى إذا كان ملف التضمين مفقودًا، فاستخدم عبارة include. غير ذلك، في حالة FrameWork، أو CMS، أو ترميز PHP للتطبيقات المعقدة، استخدم دائمًا جملة require لتضمين ملف مفتاح في تدفق التنفيذ. سيساعد ذلك في تجنب المساس بأمان تطبيقك ونزاهته، فقط في حالة فقدان ملف مفتاح واحد بطريق الخطأ.

التضمين يوفر الكثير من العمل. هذا يعني أنه يمكنك إنشاء رأس أو تذييل أو ملف قائمة قياسي لجميع صفحات الويب الخاصة بك. ثم، عندما يحتاج الرأس إلى تحديث، يمكنك فقط تحديث ملف تضمين الرأس.

بناء الجملة

```
include 'filename';

or

require 'filename';
```

أمثلة على التضمين في php

المثال الأول:

افترض أن لدينا ملف تذييل قياسي يسمى "footer.php" يبدو كالتالي:

```
<?php
echo "<p>Copyright &copy; 1999-" . date("Y") . "
W3Schools.com</p>";
?>
```

لتضمين ملف التضمين في الصفحة، استخدم جملة include:

مـثــال:

```
<html>
<body>

<h1>Welcome to my home page!</h1>
<p>Some text.</p>
<p>Some more text.</p>
<?php include 'footer.php';?>

</body>
</html>
```

المثال الثاني

افترض أن لدينا ملف قائمة قياسي يسمى "menu.php":

```
<?php
echo '<a href="/default.asp">Home</a> -
<a href="/html/default.asp">HTML Tutorial</a> -
<a href="/css/default.asp">CSS Tutorial</a> -
<a href="/js/default.asp">JavaScript Tutorial</a>
-
<a href="default.asp">PHP Tutorial</a>';
?>
```

يجب أن تستخدم جميع صفحات موقع الويب ملف القائمة هذا. فيما يلي كيفية القيام بذلك (نحن نستخدم عنصر <div> بحيث يمكن بسهولة تصميم القائمة باستخدام CSS لاحقًا):

مـــثـــال:

```
<html>
<body>

<div class="menu">
<?php include 'menu.php';?>
</div>

<h1>Welcome to my home page!</h1>
<p>Some text.</p>
<p>Some more text.</p>

</body>
</html>
```

المثال الثالث

افترض أن لدينا ملفا يسمى "vars.php"، مع تحديد بعض المتغيرات:

```
<?php
$color='red';
$car='BMW';
?>
```

ثم، إذا قمنا بتضمين ملف "vars.php" يمكن استخدام المتغيرات في ملف الاستدعاء:

مـثـــال:

```
<html>
<body>

<h1>Welcome to my home page!</h1>
<?php include 'vars.php';
echo "I have a $color $car.";
?>

</body>
</html>
```

Include مقابل require

جملة الـ require تستخدم أيضا لتضمين ملف داخل كود.

ومع ذلك، هناك فرق كبير بين include وrequire؛ عند تضمين ملف بجملة include وphp لم تتمكن من العثور عليه، البرنامج النصي سيستمر في التنفيذ:

مـــثــــال:

```
<html>
<body>

<h1>Welcome to my home page!</h1>
<?php include 'noFileExists.php';
echo "I have a $color $car.";
?>

</body>
</html>
```

في حالة قيامنا بنفس المثال باستخدام جملة require، فلن يتم تنفيذ جملة الـ echo لأن تنفيذ البرنامج النصي توقف بعد أن أرجعت جملة require خطأ فادح "fatal error":

مـــثــــال:

```
<html>
<body>

<h1>Welcome to my home page!</h1>
<?php require 'noFileExists.php';
echo "I have a $color $car.";
?>

</body>
</html>
```

❖ استخدم require عندما يكون الملف مطلوبا للتطبيق.

❖ استخدم include عندما لا يكون الملف مطلوبا ويجب أن يستمر التطبيق عند عدم العثور على الملف.

معالجة الملف

معالجة الملف جزء مهم لأي تطبيق ويب. غالبا ما تحتاج لفتح ملف ومعالجته لمهام مختلفة.

تحتوي php على العديد من الدوال لإنشاء الملفات وقراءتها وتحميلها وتحريرها.

كن حذرا عن معالجة الملفات!

يمكن أن تتسبب في ضرر كبير إذا كنت تفعل شيئا خاطئ.

الأخطاء الشائعة هي:

تحرير الملف الخطأ، ملء القرص الصلب بالبيانات المهملة، حذف محتوى الملف بالخطأ.

دالة readfile()

تقرأ الملف وتكتبه في المخزن المؤقت للإخراج.

افترض أن لدينا ملفا نصيا باسم "webdictionary.txt"، مخزنا على الخادم، يبدو كالتالي:

```
AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language
```

أكواد الـ php لقراءة الملف وكتابته للمخزن المؤقت للإخراج كما يلي (ترجع الدالة ()readfile عدد البايتات التي تمت قراءتها عند النجاح "success"):

مـــثـــال:

```
<?php
echo readfile("webdictionary.txt");
?>
```

دالة ()readfile مفيدة إذا كان كل ما تريد القيام به هو فتح ملف وقراءة محتوياته.

فتح/قراءة/إغلاق الملف

في هذا الفصل ستتعلم كيف تفتح/تقرأ/تغلق الملف من على الخادم.

فتح الملف – ()fopen

الطريقة الأفضل لفتح الملفات هي باستخدام الدالة ()fopen. تمنحك هذه الدالة خيارات أكثر من دالة ()readfile.

سنستخدم الملف النصي "webdictionary.txt"، خلال الدروس:

```
AJAX = Asynchronous JavaScript and XML
CSS = Cascading Style Sheets
HTML = Hyper Text Markup Language
PHP = PHP Hypertext Preprocessor
SQL = Structured Query Language
SVG = Scalable Vector Graphics
XML = EXtensible Markup Language
```

تحتوي المعلمة الأولى لـ ()fopen على اسم الملف الذي سيتم فتحه، وتحدد المعلمة الثانية في أي وضع يجب فتح الملف، ينشئ المثال التالي أيضا رسالة إذا كانت الدالة ()fopen غير قادرة على فتح الملف المحدد:

مــثـــال:

```
<?php
$myfile =
fopen("webdictionary.txt", "r") or die("Unabl
e to open file!");
echo fread($myfile,filesize("webdictionary.tx
t"));
fclose($myfile);
?>
```

يمكن فتح الملف في أحد الأوضاع التالية:

الوضع	الوصف
r	**فتح الملف للقراءة فقط.** يبدأ مؤشر الملف في بداية الملف
w	**فتح الملف للكتابة فقط.** مسح محتويات الملف أو انشاء ملف جديد إذا لم يكن موجودا. يبدأ مؤشر الملف في بداية الملف
a	فتح الملف للكتابة فقط. البيانات الموجودة في الملف تكون محفوظة، يبدأ مؤشر الملف من النهاية. ينشئ ملفا جديدا إذا لم يكن غير موجود
x	إنشاء ملف جديد للكتابة فقط. يرجع FALSE و an error إذا كان موجودا بالفعل.
r+	فتح ملف للـ قراءة/الكتابة. يبدأ مؤشر الملف في بداية الملف
w+	فتح الملف للقراءة/الكتابة. مسح محتويات الملف أو إنشاء ملف جديد إذا لم يكن موجودا. مؤشر الملف يبدأ من بداية الملف
a+	فتح الملف للقراءة/الكتابة. البيانات الموجودة في الملف محفوظة. مؤشر الملف يبدأ من نهاية الملف. إنشاء ملف جديد إذا كان الملف غير موجود
x+	إنشاء ملف جديد للـ قراءة/كتابة. يرجع FALSE و an error إذا كان موجودا بالفعل.

قراءة الملف – ()fread

دالة الـ ()fread تقرأ من ملف مفتوح.

تحتوي المعلمة الأولى من ()fread على اسم الملف المراد القراءة منه وتحدد المعلمة الثانية الحد الأقصى لعدد وحدات البايت المراد قراءتها.

يقرأ كود PHP التالي ملف "webdictionary.txt" حتى النهاية:

```
fread($myfile,filesize("webdictionary.txt"));
```

إغلاق الملف – ()fclose

دالة الـ ()fclose يستخدم لإغلاق ملف مفتوح.

إنها عادة برمجية جيدة لإغلاق كل الملفات بعد انتهائك منها. أنت لا تريد ملفا مفتوحا يعمل على الخادم الذي يستهلك الموارد!

تطلب ()fclose اسم الملف (أو متغير يحمل اسم الملف) الذي تريد إغلاقه:

```
<?php
$myfile = fopen("webdictionary.txt", "r");
// some code to be executed....
fclose($myfile);
?>
```

قراءة سطر واحد – ()fgets

دالة الـ ()fgets تستخدم لقراءة سطر واحد من الملف.

المثال التالي سيخرج أول سطر من ملف "webdictionary.txt":

مــثـــال:

```
<?php
$myfile =
fopen("webdictionary.txt", "r") or die("Unabl
e to open file!");
echo fgets($myfile);
fclose($myfile);
?>
```

ملاحظة: بعد استدعاء دالة الـ ()fgets، مؤشر الملف سيتحرك للسطر التالي.

تحقق من نهاية الملف – ()feof

Check End-Of-File – feof()

تقوم دالة ()feof بالتحقق مما إذا كان قد تم الوصول لنهاية الملف.

دالة الـ ()feof مفيدة للحلقات لبيانات ذات طول غير معروف.

المثال التالي يقرأ ملف الـ "webdictionary.txt" سطر سطر حتى يصل لنهاية الملف:

مــثـــال:

```
<?php
$myfile =
fopen("webdictionary.txt", "r") or die("Unabl
e to open file!");
// Output one line until end-of-file
while(!feof($myfile)) {
  echo fgets($myfile) . "<br>";
}
fclose($myfile);
?>
```

قراءة حرف واحد – fgetc()

تستخدم دالة الـ fgetc() لقراءة حرف واحد من الملف.

المثال التالي يقرأ ملف الـ "webdictionary.txt" حرف حرف، حتى يصل لنهاية الملف:

مـثـــال:

```
<?php
$myfile =
fopen("webdictionary.txt", "r") or die("Unabl
e to open file!");
// Output one character until end-of-file
while(!feof($myfile)) {
  echo fgetc($myfile);
}
fclose($myfile);
?>
```

ملاحظة: بعد استدعاء دالة الـ fgetc()، مؤشر الملف ينتقل للحرف التالي.

إنشاء/كتابة ملف

سنعلمك في هذا الفصل كيفية إنشاء ملف على الخادم والكتابة عليه.

إنشاء ملف – fopen()

دالة الـ fopen() تستخدم أيضا لإنشاء ملف. ربما مربكة بعض الشيء، ولكن في php، يتم إنشاء ملف باستخدام نفس الدالة المستخدمة لفتح الملفات.

إذا كنت تستخدم fopen() على ملف غير موجود، فسوف يقوم بإنشائه، بالنظر إلى أن الملف مفتوح للكتابة (w) أو إلحاق (a).

المثال التالي سيقوم بإنشاء ملف جديد يسمى "testfile.txt". سيتم إنشاء الملف في نفس المجلد حيث يوجد كود الـ php:

مـــثـــال:

```
$myfile = fopen("testfile.txt", "w")
```

أذونات الملف

إذا كنت تواجه أخطاء عند محاولة تشغيل هذا الكود، فتحقق من منحك حق الوصول إلى ملف الـ php لكتابة المعلومات على القرص الصلب.

الكتابة على الملف – ()fwrite

دالة الـ ()fwrite تستخدم للكتابة على الملفات.

المعلمة الأولى لدالة ()fwrite تحتوي على اسم الملف المراد الكتابة عليه والمعلمة الثانية للنص المراد كتابته.

المثال التالي سيكتب بضعة أسماء في ملف جديد يسمى "newfile.txt":

مـــثـــال:

```
<?php
$myfile =
fopen("newfile.txt", "w") or die("Unable to
open file!");
$txt = "John Doe\n";
fwrite($myfile, $txt);
$txt = "Jane Doe\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

لاحظ أننا كتبنا على ملف "newfile.txt" مرتين. في كل مرة كتبنا فيها على الملف ارسلنا السلسلة النصية `$txt` الأولى التي تحتوي على "john Doe" والثانية تحتوي على "Jane Doe".

بعدما انتهينا من الكتابة، سنغلق الملف باستخدام دالة الـ `fclose()`.

إذا فتحنا ملف الـ "newfile.txt" سيبدوا كالتالي:

```
John Doe
Jane Doe
```

الكتابة فوق (Overwriting)

الان ملف "newfile.txt" يحتوي على بعض البيانات، يمكننا رؤية ماذا يحدث عندما نفتح الملف الموجود للكتابة. سيتم مسح جميع البيانات الحالية ونبدأ بملف فارغ.

في المثال التالي سنفتح ملف "newfile.txt"، ونكتب بعض البيانات الجديدة داخله:

مــثـــال:

```php
<?php
$myfile =
fopen("newfile.txt", "w") or die("Unable to
open file!");
$txt = "Mickey Mouse\n";
fwrite($myfile, $txt);
$txt = "Minnie Mouse\n";
fwrite($myfile, $txt);
fclose($myfile);
?>
```

إذا فتحنا ملف الـ "newfile.txt"، اختفت أسماء john و jane، ووجدنا البيانات التي كتبناها الأن فقط:

```
Mickey Mouse
Minnie Mouse
```

رفع ملفات php

باستخدام php، من السهل تحميل الملفات على الخادم.

ومع ذلك، وبالسهولة يأتي الخطر، لذلك كن دائما حذرا عند السماح بتحميل الملفات!

تكوين ملف "php.ini"

أولا، تأكد من ضبط php للسماح بتحميل الملفات.

في ملف "php.ini" ابحث عن تعليمات `file_uploads`، واضبطه على On:

انشاء نموذج html

بعد ذلك، قم بإنشاء نموذج html يتيح للمستخدمين اختيار ملف الصورة الذي يريدون تحميله:

مـــثـــال:

```
<!DOCTYPE html>
<html>
<body>

<form action="upload.php" method="post" encty
pe="multipart/form-data">
    Select image to upload:
    <input type="file" name="fileToUpload" id
="fileToUpload">
    <input type="submit" value="Upload
Image" name="submit">
</form>

</body>
</html>
```

بعض القواعد الواجب اتباعها لنموذج html السابق:

- تأكد من أن النموذج يستخدم method="post".
- يحتاج النموذج أيضا للخاصية التالية: enctype="multipart/form-data". وهي لتحديد نوع المحتوى الذي يجب استخدامه عند إرسال النموذج.

بدون المتطلبات السابقة، لن يعمل تحميل الملفات.

ملاحظات على أشياء أخرى:

خاصية الـ `type="file"` لوسم الـ `<input>` تعرض حقل الإدخال كعنصر تحكم لتحديد الملفات، مع وجود زر "Browse" بجوار عنصر تحكم الإدخال.

النموذج السابق يرسل البيانات لملف يسمى "upload.php" والذي سنقوم بإنشائه بعد ذلك.

إنشاء ملف تحميل

يحتوي ملف الـ "upload.php" على كود تحميل الملفات:

مــثـــال:

```
<?php
$target_dir = "uploads/";
$target_file = $target_dir .
basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType
= strtolower(pathinfo($target_file,PATHINFO_E
XTENSION));
// Check if image file is a actual image or
fake image
if(isset($_POST["submit"])) {
    $check =
getimagesize($_FILES["fileToUpload"]["tmp_nam
e"]);
    if($check !== false) {
        echo "File is an image - " .
$check["mime"] . ".";
        $uploadOk = 1;
    } else {
        echo "File is not an image.";
        $uploadOk = 0;
    }
}
?>
```

توضيح البرنامج النصي:

- ❖ `$target_dir = "uploads/"`

يحدد المجلد الذي سيتم وضع الملف فيه.

- ❖ `$target_file`

يحدد مسار الملف المراد تحميله

- ❖ `$uploadOk=1`

لم يتم استخدامه بعد (سيتم استخدامه لاحقا)

- ❖ `$imageFileType`

يحمل امتداد الملف للملف (بالأحرف الصغيرة)

بعد ذلك، تحقق مما إذا كان ملف الصورة صورة حقيقية أم صورة مزيفة.

ملاحظة: ستحتاج لإنشاء مجلد جديد يسمى "uploads" في المجلد الذي يوجد فيه ملف "upload.php". الملفات المحملة ستحفظ هناك.

تحقق مما إذا الملف موجودا بالفعل

الأن يمكننا إضافة بعض القيود.

أولا، سنتحقق مما إذا كان مجلد "uploads" موجودا بالفعل. إذا كان موجود، ستظهر رسالة خطأ، و $uploadOk سيتغير لـ 0:

```
// تحقق مما إذا كان الملف موجود بالفعل
if (file_exists($target_file)) {
    echo "عذرا، الملف موجود بالفعل!";
    $uploadOk = 0;
}
```

شرط حجم الملف

يسمى حقل إدخال الملف في نموذج HTML السابق باسم "fileToUpload".

الأن، نريد التحقق من حجم الملف. إذا كان الملف أكبر من 500 كيلو بايت، فسيتم عرض رسالة الخطأ، وتعيين `$uploadOk` على 0:

```
// Check file size
if ($_FILES["fileToUpload"]["size"] >
500000) {
    echo "Sorry, your file is too large.";
    $uploadOk = 0;
}
```

شرط نوع الملف

يسمح الكود التالي للمستخدمين فقط بتحميل ملفات JPG و JPEG و PNG و GIF.

وكل أنواع الملفات الأخرى ستظهر رسالة خطأ قبل تعيين `$uploadOk` لـ 0:

مــثـــال:

```
// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType
!= "png" && $imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
    echo "Sorry, only JPG, JPEG, PNG & GIF
files are allowed.";
    $uploadOk = 0;
}
```

البرنامج النصي الكامل لتحميل الملفات

يبدو ملف "upload.php" الكامل كالتالي:

```
<?php
$target_dir = "uploads/";
$target_file = $target_dir .
basename($_FILES["fileToUpload"]["name"]);
$uploadOk = 1;
$imageFileType
= strtolower(pathinfo($target_file,PATHINFO_E
XTENSION));
// تحقق من أن الصورة حقيقة وليست مزيفة
if(isset($_POST["submit"])) {
    $check =
getimagesize($_FILES["fileToUpload"]["tmp_nam
e"]);
    if($check !== false) {
        echo "File is an image - " .
$check["mime"] . ".";
        $uploadOk = 1;
    } else {
        echo "File is not an image.";
        $uploadOk = 0;
    }
}
// التحقق من أن الملف موجود بالفعل
if (file_exists($target_file)) {
    echo "Sorry, file already exists.";
    $uploadOk = 0;
}
// التحقق من حجم الملف
```

```
if ($_FILES["fileToUpload"]["size"] > 500000)
{
    echo "Sorry, your file is too large.";
    $uploadOk = 0;
}
// Allow certain file formats
if($imageFileType != "jpg" && $imageFileType
!= "png" && $imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
    echo "Sorry, only JPG, JPEG, PNG & GIF
files are allowed.";
    $uploadOk = 0;
}
// Check if $uploadOk is set to 0 by an error
if ($uploadOk == 0) {
    echo "Sorry, your file was not
uploaded.";
// إذا كان كل شيء جيد؛ حاول رفع الملف //
} else {
    if (move_uploaded_file($_FILES["fileToUpl
oad"]["tmp_name"], $target_file)) {
        echo "The file ".
basename( $_FILES["fileToUpload"]["name"]). "
has been uploaded.";
    } else {
        echo "Sorry, there was an error
uploading your file.";
    }
}
?>
```

ماهي الـ Cookie؟

غالبًا ما يتم استخدام ملف تعريف الارتباط لتعريف المستخدم. ملف تعريف الارتباط هو ملف صغير يقوم الخادم بتضمينه على حاسوب المستخدم. في كل مرة يطلب فيها الحاسوب نفسه صفحة مع متصفح، فإنه سيرسل ملف تعريف الارتباط أيضًا. باستخدام PHP، يمكنك إنشاء واسترجاع قيم ملفات تعريف الارتباط.

إنشاء الـ Cookie باستخدام php

الـ cookie تنشأ باستخدام دالة الـ `setcookie()`.

بناء الجملة

```
setcookie(name, value, expire, path, domain, secure,
httponly);
```

معلمة الاسم فقط مطلوبة، أما الباقي كلها اختيارية.

إنشاء/ استرداد ملف تعريف الارتباط

المثال التالي سينشئ cookie باسم "user" بقيمة "John Doe".

الـ cookie ستنتهي صلاحيتها بعد ٣٠ يوم (86400*30). الـ "/" تعني ملف تعريف الإرتباط متاح في موقع الويب بالكامل (وإلا حدد المجلد الذي تفضله).

ثم نقوم باسترداد قيمة الـ cookie "user" (باستخدام المتغير العالمي `$_COOKIE`). سنستخدم أيضا دالة `isset()` لمعرفة ما إذا كان الـ cookie معينا أم لا:

مـــثـــال:

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value,
time() + (86400 * 30), "/"); // 86400 = 1
day
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name
. "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is
set!<br>";
    echo "Value is:
```

```
" . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

ملاحظات:

- دالة الـ `setcookie()` يجب أن تظهر قبل وسم `<tag>`.
- يتم تشفير قيمة الـ cookie تلقائيًا عند إرساله، ويتم فك تشفيره تلقائيًا عند استلامه (لمنع URLencoding، استخدم `setrawcookie()` بدلاً من ذلك).

تعديل قيمة ملف تعريف الإرتباط

لتعديل ملف تعريف الارتباط، فقط قم بتعيين (again) ملف تعريف الإرتباط باستخدام دالة :setcookie()

مـــثـــال:

```
<?php
$cookie_name = "user";
$cookie_value = "Alex Porter";
setcookie($cookie_name, $cookie_value,
time() + (86400 * 30), "/");
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
    echo "Cookie named '" . $cookie_name
. "' is not set!";
} else {
    echo "Cookie '" . $cookie_name . "' is
set!<br>";
    echo "Value is:
" . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

حذف ملفات تعريف الإرتباط Cookie

لحذف ملفات تعريف الإرتباط، استخدم دالة `setcookie()` بتعين تاريخ انتهاء الصلاحة في الماضي:

مـــثـــال:

```
<?php
// set the expiration date to one hour ago
setcookie("user", "", time() - 3600);
?>
<html>
<body>

<?php
echo "Cookie 'user' is deleted.";
?>

</body>
</html>
```

تحقق مما إذا كان تعريف الإرتباط ممكنا

يقوم المثال التالي بإنشاء برنامج نصي صغير يتحقق من تمكين ملفات تعريف الإرتباط.

أولا، حاول إنشاء ملف تعريف الارتباط للاختبار باستخدام دالة `setcookie()`، ثم احسب متغير المصفوفة `$_COOKIE`:

مــثـــال:

```
<?php
setcookie("test_cookie", "test", time()
+ 3600, '/');
?>
<html>
<body>

<?php
if(count($_COOKIE) > 0) {
    echo "Cookies are enabled.";
} else {
    echo "Cookies are disabled.";
}
?>

</body>
</html>
```

جلسات php

الجلسة هي وسيلة لتخزين المعلومات (في المتغيرات) لاستخدامها عبر صفحات متعددة.

على عكس الـ cookie، لا يتم تخزين المعلومات على حاسوب المستخدم.

ماهي جلسات php؟

عند العمل مع أحد التطبيقات، تقوم بفتحه، وتعمل بعض التغييرات، ثم تقوم بإغلاقه. هذا يشبه إلى حد كبير الجلسة. الحاسوب يعرف من أنت. يعرف متى تبدأ التطبيق ومتى تنتهي. ولكن هناك مشكلة واحدة على الإنترنت: خادم الويب لا يعرف من أنت أو ماذا تفعل، لأن عنوان HTTP لا يحافظ على حالته.

متغيرات الجلسة تحل هذه المشكلة عن طريق تخزين معلومات المستخدم لاستخدامها عبر صفحات متعددة (مثل اسم المستخدم أو اللون المفضل، إلخ). بشكل افتراضي، تستمر متغيرات الجلسة حتى يغلق المستخدم المتصفح.

وبالتالي؛ تحتوي متغيرات الجلسة معلومات عن مستخدم واحد، وتتوفر لجميع الصفحات في تطبيق واحد.

ملاحظة: إذا كنت بحاجة للتخزين الدائم، فقد ترغب في تخزين البيانات في قاعدة بيانات.

بدء جلسة php

تبدأ الجلسة باستخدام دالة `session_start()`.

يتم تعيين متغيرات الجلسة بمغير php العالمي `$_SESSION`.

الأن، دعونا ننشئ صفحة جديدة تسمى "demo_session1.php". في هذه الصفحة سنبدأ جلسة جديدة ونضع بعض متغيرات الجلسة:

مـثــال:

```
<?php
// بدء الجلسة
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// وضع قيم المتغيرات للجلسة
$_SESSION["favcolor"] = "green";
$_SESSION["favanimal"] = "cat";
echo "Session variables are set.";
?>

</body>
</html>
```

ملاحظة: دالة الـ `session_start()` يجب أن تكون أول شي في مستندك. قبل أي وسم من أوسمة html.

جلب قيم متغيرات الجلسة

بعد ذلك، نقوم بإنشاء صفحة أخرى تسمى "demo_session2.php". من هذه الصفحة، سوف نصل لمعلومات الجلسة التي حددناها في الصفحة الأولى ("demo_session1.php").

لاحظ أنه لا يتم تمرير متغيرات الجلسة بشكل فردي لكل صفحة جديدة، بل يتم استردادها من الجلسة التي نفتحها في بداية كل صفحة (`session_start()`).

لاحظ أيضا أنه يتم تخزين جميع قيم متغير الجلسة في المتغير العالمي `$_SESSION`:

مـــثـــال:

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// Echo session variables that were set on
previous page
echo "Favorite color is
" . $_SESSION["favcolor"] . ".<br>";
```

```
echo "Favorite animal is
" . $_SESSION["favanimal"] . ".";
?>

</body>
</html>
```

هناك طريقة أخرى لإظهار جميع قيم متغير الجلسة لجلسة المستخدم وهي تشغيل الكود التالي:

مـــثـــال:

```

<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
print_r($_SESSION);
?>

</body>
</html>
```

كيف تعمل؟ كيف عرفت أنه أنا؟

تقوم معظم الجلسات بتعيين مفتاح مستخدم على جهاز الحاسوب الخاص بالمستخدم يبدو كالتالي:

765487cf34ert8dede5a562e4f3a7e12

ثم عندما يتم فتح جلسة على صفحة أخرى، فإنه يقوم بمسح الحاسوب بحثا عن مفتاح مستخدم. إذا كان هناك تطابق، فإنه يصل لتلك الجلسة، وإذا لم يكن كذلك، فإنه يبدأ جلسة جديدة.

تعديل متغير الجلسة

لتغير متغير الجلسة، فقط اكتبه فوقه:

مــثـــال:

```
<?php session_start(); ?>
<!DOCTYPE html>
<html> <body>
<?php
// لتغيير متغير الجلسة، فقط اكتبه فوقه
$_SESSION["favcolor"] = "yellow";
print_r($_SESSION);
?>
</body> </html>
```

تدمير أو إغلاق الجلسة

لمسح كل متغيرات الجلسة العالمية وتدمير الجلسة، استخدم `session_unset()` و `session_destroy()`:

مـــثـــال:

```
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// remove all session variables
session_unset();

// destroy the session
session_destroy();
?>

</body>
</html>
```

مرشحات php

التحقق من صحة البيانات "Validating data" = تحديد ما إذا كانت البيانات في شكلها الصحيح.

تعقيم البيانات "Sanitizing data" = إزالة أي حروف غير قانونية من البيانات.

ملحقات مرشحات php

تستخدم مرشحات php للتحقق من صحة المدخلات الخارجية وتعقيمها.

تحتوي ملحق مرشح php على العديد من الدوال اللازمة لفحص إدخال المستخدم، وهو مصمم لجعل التحقق من صحة البيانات أسهل وأسرع.

دالة الـ `filter_list()` يمكن استخدامها لسرد ما يقدمه امتداد مرشح php:

مــثـــال:

```
<table>
  <tr>
    <td>Filter Name</td>
    <td>Filter ID</td>
  </tr>
  <?php
  foreach (filter_list() as $id =>$filter) {
      echo '<tr><td>' . $filter
. '</td><td>' . filter_id($filter)
. '</td></tr>';
  }
  ?>
</table>
```

لماذا نستخدم المرشحات؟

تتلقى العديد من تطبيقات الويب مدخلات خارجية، يمكن أن تكون مدخلات/ بيانات خارجية:

- ❖ إدخال المستخدم من نموذج.
- ❖ ملفات تعريف الإرتباط.
- ❖ بيانات خدمات الويب.
- ❖ متغيرات الخادم.
- ❖ نتائج استعلام قاعدة البيانات.

يجب عليك دائما التحقق من صحة البيانات الخارجية!
البيانات المقدمة غير الصالحة يمكن أن تؤدي لمشاكل أمنية وتعطيل صفحة الويب الخاصة بك!
باستخدام المرشحات يمكنك التأكد من حصول التطبيق على المدخلات الصحيحة!

دالة `filter_var()`

تعمل دالة الـ `filter_var()` على التحقق من صحة البيانات وتعقيمها.

تقوم دالة الـ `fileter_var()` بتصفية متغير واحد بمشرح محدد يأخذ قطعتين من البيانات:

- ❖ المتغير الذي تريد التحقق منه
- ❖ نوع التحقق المستخدم

تعقيم السلاسل النصية

المثال التالي يستخدم دالة الـ `filter_var()` لإزالة كل أوسمة الـ html من السلسلة:

مـثـــال:

```
<?php
$str = "<h1>Hello World!</h1>";
$newstr = filter_var($str,
FILTER_SANITIZE_STRING);
echo $newstr;
?>
```

التحقق من صحة العدد الصحيح

المثال التالي يستخدم دالة الـ `filter_var()` للتحقق من صحة المتغير `$int` هل هو عدد صحيح أم لا، إذا كان المتغير عدد صحيح: فسيكون ناتج الكود التالي: "Integer is valid". إذا كان المتغير ليس عددا صحيحا، فسيكون الناتج أو المخرج: "Integer is not valid":

مـثـــال:

```
<?php
$int = 100;

if (!filter_var($int, FILTER_VALIDATE_INT)
=== false) {
    echo("Integer is valid");
} else {
```

```
    echo("Integer is not valid");
}
?>
```

ملاحظة: مشكلة الدالة `filter_var()` مع الصفر

في المثال السابق، إذا تم تعيين المتغير على 0 فسوف ترجع الدالة "Integer is not valid". لحل هذه المشكلة؛ استخدم الكود التالي:

مــثـــال:

```
<?php
$int = 0;

if (filter_var($int, FILTER_VALIDATE_INT)
=== 0 || !filter_var($int,
FILTER_VALIDATE_INT) === false) {
    echo("Integer is valid");
} else {
    echo("Integer is not valid");
}
?>
```

التحقق من صحة عنوان IP

في المثال التالي سنستخدم دالة `filter_var()` للتحقق مما إذا كان المتغير `$ip` هو عنوان ip صالح:

مــثـــال:

```
<?php
$ip = "127.0.0.1";

if (!filter_var($ip, FILTER_VALIDATE_IP) ===
false) {
    echo("$ip is a valid IP address");
} else {
    echo("$ip is not a valid IP address");
}
?>
```

تعقيم والتحقق من صحة عنوان البريد الإلكتروني

في المثال التالي سنستخدم الدالة `filter_var()` لإزالة جميع الأحرف غير القانونية أولا من متغير `$email`، ثم تحقق مما إذا كان عنوان البريد الإلكتروني صالحا:

مــثـــال:

```
<?php
$email = "john.doe@example.com";

// Remove all illegal characters from email
$email = filter_var($email,
FILTER_SANITIZE_EMAIL);

// Validate e-mail
if (!filter_var($email,
FILTER_VALIDATE_EMAIL) === false) {
    echo("$email is a valid email address");
} else {
    echo("$email is not a valid email
address");
}
?>
```

تطهير والتحقق من صحة عنوان URL

في المثال التالي سنستخدم دالة الـ `filter_var()` للمسح الأولي لكل الحروف الغير قانونية من عنوان الـURL، ثم تحقق مما إذا كان `$url` هو عنوان url صالح:

مـــثـــال:

```
<?php
$url = "https://www.w3schools.com";

// Remove all illegal characters from a url
$url = filter_var($url,
FILTER_SANITIZE_URL);

// Validate url
if (!filter_var($url, FILTER_VALIDATE_URL)
=== false) {
    echo("$url is a valid URL");
} else {
    echo("$url is not a valid URL");
}
?>
```

مرشحات php متقدمة

التحقق من صحة العدد الصحيح داخل النطاق.

في المثال التالي سنستخدم دالة الـ `filter_var()` للتحقق مما إذا كان المتغير كلاهما من النوع الصحي، و بين ١ و ٢٠٠:

مــثـــال:

```
<?php
$int = 122;
$min = 1;
$max = 200;

if (filter_var($int,
FILTER_VALIDATE_INT, array("options" => array
("min_range"=>$min, "max_range"=>$max))) ===
false) {
    echo("Variable value is not within the
legal range");
} else {
    echo("Variable value is within the legal
range");
}
?>
```

التحقق من عنوان IPv6

المثال التالي سيستخدم دالة الـ `filter_var()` للتحقق مما إذا كان المتغير `$ip` هو عنوان IPv6 صالح:

مـثـــال:

```
<?php
$ip
= "2001:0db8:85a3:08d3:1319:8a2e:0370:7334";

if (!filter_var($ip, FILTER_VALIDATE_IP,
FILTER_FLAG_IPV6) === false) {
    echo("$ip is a valid IPv6 address");
} else {
    echo("$ip is not a valid IPv6 address");
}
?>
```

التحقق من صحة عنوان URL – يجب أن يحتوي على QueryString

المثال التالي سيستخدم دالة الـ `filter_var()` للتحقق من المتغير `$url` هو هو عنوان URL بـ querystring:

مــثـــال:

```
<?php
$url = "https://www.w3schools.com";

if (!filter_var($url, FILTER_VALIDATE_URL,
FILTER_FLAG_QUERY_REQUIRED) === false) {
    echo("$url is a valid URL with a query
string");
} else {
    echo("$url is not a valid URL with a
query string");
}
?>
```

مسح الحروف بقيمة الـ ASCII > 127

في المثال التالي سنستخدم الدالة `filter_var()` لتطهير السلسلة النصية، كلاهما سيمسح أوسمة html و كل أحرف بقيمة ASCII value > 127 من السلسلة:

مــثـــال:

```
<?php
$str = "<h1>Hello WorldÆØÅ!</h1>";

$newstr = filter_var($str,
FILTER_SANITIZE_STRING,
FILTER_FLAG_STRIP_HIGH);
echo $newstr;
?>
```

php و JSON

ما هو الـ JSON؟

أولا هو اختصار لـ "javaScript Object Notation"، وهو عبارة عن بناء جملة لتخزين وتبادل الملفات.

نظرا لأن تنسيق json هو تنسيق يستند على نص، يمكن إرساله بسهولة من وإلى الخادم، واستخدامه كتنسيق بيانات بواسطة أي لغة برمجة.

Php و json يحتوي على بعض الدوال المدمجة للتعامل معها.

php: أولا، سينشئ الدوال التالية:

- json_encode()
- json_decode()

دالة `json_encode()`

هذه الدالة تستخدم لتشفير قيم تنسيق JSON.

مــثـــال:

يوضح هذا المثال كيفية تشفير مصفوفة اقتران في كائن JSON:

```
<?php
$age
= array("Peter"=>35, "Ben"=>37, "Joe"=>43);

echo json_encode($age);
?>
```

مــثـــال:

هذا المثال يبين كيفية تشفير مصفوفة مفهرسة في مصفوفة JSON:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");

echo json_encode($cars);
?>
```

دالة `json_decode()`

هذه الدالة تستخدم لفك تشفير كان الـ json داخل كائن php أو مصفوفة متصلة.

مـــثـــال:

هذا المثال يقوم بفك تشفير بيانات json لكائن php:

```
<?php
$jsonobj = '{"Peter":35,"Ben":37,"Joe":43}';

var_dump(json_decode($jsonobj));
?>
```

ترجع دالة `json_decode()` كائنا افتراضيا. هذه الدالة على معلمة ثانية، وعندما يتم تعيينها على true يتم فك تشفير كائنات json في مصفوفات مقترنة.

مـــثـــال:

في هذا المثال نقوم بفك تشفير البيانات في مصفوفة:

```
<?php
$jsonobj = '{"Peter":35,"Ben":37,"Joe":43}';

var_dump(json_decode($jsonobj, true));
?>
```

الوصول للقيم المشفرة

فيما يلي مثالان لكيفية الوصول إلى القيم التي تم فك تشفيرها من كائن ومن مصفوفة:

مــثــــال:

هذا المثال يعرض كيفية الوصول للقيم من كائن php:

```
<?php
$jsonobj = '{"Peter":35,"Ben":37,"Joe":43}';

$obj = json_decode($jsonobj);

echo $obj->Peter;
echo $obj->Ben;
echo $obj->Joe;
?>
```

مـثـــال:

يوضح هذا المثال كيفية الوصول للقيم من مصفوفة:

```
<?php
$jsonobj = '{"Peter":35,"Ben":37,"Joe":43}';

$arr = json_decode($jsonobj, true);

echo $arr["Peter"];
echo $arr["Ben"];
echo $arr["Joe"];
?>
```

حلقات من خلال القيم

يمكنك أيضا تنفيذ حلقة من خلال القيم باستخدام حلقة `foreach()`:

مـثـــال:

يوضح هذا المثال كيفية تنفيذ حلقات كائن php:

```
<?php
$jsonobj = '{"Peter":35,"Ben":37,"Joe":43}';
$obj = json_decode($jsonobj);
foreach($obj as $key => $value) {
  echo $key . " => " . $value . "<br>";
}
?>
```

مــثـــال:

يوضح هذا المثال كيفية تنفيذ حلقات المصفوفة:

```
<?php
$jsonobj = '{"Peter":35,"Ben":37,"Joe":43}';

$arr = json_decode($jsonobj, true);

foreach($arr as $key => $value) {
  echo $key . " => " . $value . "<br>";
}
?>
```

قواعد بيانات MySQL

باستخدام php يمكنك الاتصال بقواعد البيانات ومعالجتها.

MySQL هو نظام قواعد البيانات الأكثر استخداما مع php.

ما هو MySQL؟

- هو نظام قواعد بيانات يستخدم على شبكة الانترنت.
- هو نظام قواعد بيانات يعمل على الخادم.
- مثالية لكل من التطبيقات الصغيرة والكبيرة.
- سريعة جدا وموثوقة وسهلة الإستخدام.
- يستخدم SQL القياسية.
- يترجم على عدد من المنصات
- مجاني للتنزيل والاستخدام
- تم تطويرها وتوزيعها ودعمها بواسطة شركة Oracle Corporation
- تم تسميتها على اسم ابنة مؤسس شركة **M**onty Widenius's daughter: My.

يتم تخزين البيانات في قاعدة بيانات MySQL في الجداول. الجداول عبارة عن مجموعة من البيانات ذات الصلة، وتتكون من أعمدة وصفوف.

قواعد البيانات مفيدة لتخزين المعلومات بشكل خاص، قد يكون لدى الشركة قاعدة بيانات تحتوي على الجداول التالية:

- ❖ الموظفين
- ❖ المنتجات
- ❖ الزبائن

نظام قواعد بيانات php + MySQL

php مع MySQL عبارة عن منصات مشتركة (يمكنك التطوير في windows والعمل على خادم على نظام تشغيل unix).

استعلامات قاعدة البيانات

الاستعلام هو سؤال أو طلب.

يمكننا الاستعلام عن قاعدة بيانات للحصول على معلومات محددة وإرجاع مجموعة سجلات.

انظر للاستعلام التالي (باستخدام SQL قياسية):

```
SELECT LastName FROM Employees
```

يحدد الإستعلام السابق كافة البيانات الموجودة في عمود "LastName" من جدول "Employees".

تنزيل قواعد بيانات MySQL

إذا لم يكن خادم php به قاعدة بيانات MySQL، فيمكنك تنزيله مجانا من هنا:

http://www.mysql.com

حقائق حول قاعدة بيانات MySQL

هو نظام قاعدة بيانات قياسي لمواقع الويب مع أحجام ضخمة من البيانات والمستخدمين النهائيين (مثل: الفيس بوك وتويتر وويكيبيديا).

شيء رائع عن MySQL هو أنه يمكن تصغيره لدعم تطبيقات قواعد البيانات المضمنة.

انظر : http://www.mysql.com/customers للحصول على نظرة عامة على الشركات التي تستخدم MySQL.

وصل php بـ MySQL

يمكن لـ php5 وما بعدها العمل بقاعدة بيانات MySQL باستخدام:

امتداد MySQLI ("I" تعني التحسين improved).

PDO (كائنات بيانات php).

تستخدم الإصدارات السابقة من php امتداد MySQL. ومع ذلك، تم إهمال هذا الامتداد في عام ٢٠١٢.

هل يجب استخدام MySQLI و PDO؟

إذا كنت بحاجة لإجابة قصيرة، فستكون "ما تريد".

تتمتع كل من MySQLI و PDO بمزاياها:

ستعمل PDO على ١٢ نظاما مختلفا لقواعد البيانات، بينما تعمل MySQLI على قواعد بيانات MySQL فقط.

لذا، إذا كان عليك تبديل مشروعك لاستخدام قاعدة بيانات أخرى، فإن PDO تجعل العملية سهلة. ما عليك سوى تغيير سلسلة الإتصال وبعض الاستعلامات، باستخدام MySQLI، ستحتاج لإعادة كتابة التعليمات البرمجية بالكامل – طلب البحث المضمنة.

كلاهما موجه نحو الكائن، لكن MySQLI يقدم أيضا واجهة برمجة تطبيقات إجرائية.

كلاهما يدعمان البيانات المعدة، البيانات المعدة تحمي من حقن SQL، وهي مهمة جدا لأمان تطبيق الويب.

MySQL Examples in Both MySQLi and PDO Syntax

في هذا الفصل والفصول التالية نعرض ثلاث طرق للعمل مع php و MySQL:

- MySQLi (object-oriented)
- MySQLi (procedural)
- PDO

تثبيت MySQLi

للوندوز واللينكس: يتم تثبيت ملحق MySQLi تلقائيًا في معظم الحالات، عند تثبيت حزمة php5 mysql.

للحصول على تفاصيل التثبيت، انتقل إلى:

http://php.net/manual/ar/mysqli.installation.php

تثبيت PDO

للحصول على تفاصيل التثبيت؛ انتقل إلى:

http://php.net/manual/en/pdo.installation.php

فتح اتصال لـ MySQL

قبل أن نتمكن من الوصول للبيانات في قاعدة البيانات نحتاج لأن نكون قادرين على الاتصال بالخادم:

مـثـــال:

(MySQLi Object-Oriented)

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// إنشاء اتصال //
$conn = new mysqli($servername, $username,
$password);

// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn-
>connect_error);
}
echo "Connected successfully";
?>
```

ملاحظة على object-oriented في المثال السابق:
تم كسر $connect_error حتى PHP 5.2.9 و5.3.0. إذا كنت بحاجة إلى ضمان التوافق مع إصدارات PHP قبل الإصدار 5.2.9 و 5.3.0، فاستخدم الكود التالي:

```
// التحقق من الإتصال
if (mysqli_connect_error()) {
    die("Database connection failed: " .
mysqli_connect_error());
}
```

مـثـــال (MySQLi Procedural):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// إنشاء اتصال
$conn = mysqli_connect($servername,
$username, $password);

// التحقق من الإتصال
if (!$conn) {
    die("Connection failed: " .
mysqli_connect_error());
}
```

```
echo "Connected successfully";
?>
```

مــثـــال (PDO):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

try {
    $conn
= new PDO("mysql:host=$servername;dbname=myDB
", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
    echo "Connected successfully";
    }
catch(PDOException $e)
    {
    echo "Connection failed: " . $e-
>getMessage();
    }
?>
```

ملاحظة: في مثال PDO السابق، حددنا أيضًا قاعدة بيانات (myDB). تتطلب PDO قاعدة بيانات صالحة للاتصال بها. إذا لم يتم تحديد قاعدة بيانات، يتم طرح استثناء.

نصيحة: تتمثل فائدة كبيرة لـ PDO في أن لديها فئة استثناء للتعامل مع أي مشاكل قد تحدث في استعلامات قاعدة البيانات الخاصة بنا. إذا تم طرح استثناء داخل كتلة `try{}`، يتوقف البرنامج النصي عن التنفيذ ويتدفق مباشرة إلى الكتلة `catch(){}` الأولى.

إغلاق الاتصال

سيتم إغلاق الاتصال تلقائيًا عند انتهاء البرنامج النصي. لإغلاق الاتصال من قبل، استخدم التالي:

MySQLi Object-Oriented:

```
$conn->close();
```

MySQLi Procedural:

```
mysqli_close($conn);
```

PDO:

```
$conn = null;
```

إنشاء قاعدة بيانات MySQL

تتكون قاعدة البيانات من جدول واحد أو أكثر.

ستحتاج إلى امتيازات CREATE خاصة لإنشاء أو حذف قاعدة بيانات MySQL.

إنشاء قاعدة بيانات MySQL باستخدام MySQLi وPDO

يتم استخدام عبارة CREATE DATABASE لإنشاء قاعدة بيانات في MySQL.

الأمثلة التالية تنشئ قاعدة بيانات باسم "myDB":

مـثـــال:

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// إنشاء اتصال
$conn = new mysqli($servername, $username,
$password);
// تحقق من الاتصال
if ($conn->connect_error) {
    die("Connection failed: " . $conn-
```

```
>connect_error);
}

// إنشاء قاعدة بيانات
$sql = "CREATE DATABASE myDB";
if ($conn->query($sql) === TRUE) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " .
$conn->error;
}

$conn->close();
?>
```

ملاحظة: عند إنشاء قاعدة بيانات جديدة، يجب عليك فقط تحديد المدخلات الثلاث الأولى لكائن mysqli (اسم الخادم واسم المستخدم وكلمة المرور).

نصيحة: إذا كان عليك استخدام منفذ معين، فأضف سلسلة فارغة لمدخل اسم قاعدة البيانات، مثل:

```
new mysqli("localhost", "username", "password",
"", port)
```

مــثـــال (MySQLi Procedural) :

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

// إنشاء إتصال //
$conn = mysqli_connect($servername,
$username, $password);
// التحقق من الاتصال //
if (!$conn) {
    die("Connection failed: " .
mysqli_connect_error());
}

// إنشاء قاعدة بيانات //
$sql = "CREATE DATABASE myDB";
if (mysqli_query($conn, $sql)) {
    echo "Database created successfully";
} else {
    echo "Error creating database: " .
mysqli_error($conn);
}

mysqli_close($conn);
?>
```

ملاحظة: يقوم المثال التالي بإنشاء قاعدة بيانات PDO باسم "myDBPDO":

مــثـــال (PDO):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";

try {
    $conn
= new PDO("mysql:host=$servername",
$username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
    $sql = "CREATE DATABASE myDBPDO";
    // use exec() because no results are
returned
    $conn->exec($sql);
    echo "تم إنشاء قاعدة بيانات بنجاح <br>";
    }
catch(PDOException $e)
    {
    echo $sql . "<br>" . $e->getMessage();
    }
```

```
$conn = null;
?>
```

نصيحة: تتمثل فائدة كبيرة لـ PDO في أن لديها فئة استثناء لمعالجة أي مشاكل قد تحدث في استعلامات قاعدة البيانات الخاصة بنا. إذا تم طرح استثناء داخل كتلة `try{}`، يتوقف البرنامج النصي عن التنفيذ ويتدفق مباشرة إلى الكتلة `catch(){}` الأولى. في كتلة `catch` السابقة نطبع عبارة SQL ورسالة الخطأ التي تم إنشاؤها.

انشاء جدول MySQL

جدول قاعدة البيانات له اسم فريد خاص به ويتكون من أعمدة وصفوف.

إنشاء جدول MySQL باستخدام MySQLi و PDO

يتم استخدام جملة CREATE TABLE لإنشاء جدول في MySQL.

سننشئ جدولا باسم "MyGuests" بخمسة أعمدة:

"id", "firstname", "lastname", "email" and "reg_date":

```
CREATE TABLE MyGuests
(
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY
KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
ON UPDATE CURRENT_TIMESTAMP
)
```

ملاحظات على الجدول السابق:

يحدد data type نوع البيانات التي يمكن أن يحتفظ بها العمود. للحصول على مرجع كامل لجميع أنواع البيانات المتاحة.

بعد نوع البيانات، يمكنك تحديد خاصيات اختيارية أخرى لكل عمود:

- ❖ NOT NULL :

يجب أن يحتوي كل صف على قيمة لهذا العمود، ولا يسمح بالقيم الخالية.

- ❖ DEFAULT value :

عين قيمة افتراضية تضاف عندما لا يتم تمرير أي قيمة أخرى.

- ❖ UNSIGNED

يستخدم لأنواع الأرقام، ويقصر البيانات المخزنة على الأرقام الموجبة والصفر.

- ❖ AUTO INCREMENT:

يزيد تلقائيا قيمة الحقل بمقدار ١ في كل مرة يتم فيها إضافة سجل جديد.

- ❖ PRIMARY KEY:

تستخدم لتحديد الصفوف في جدول فريد، غالبا ما يكون العمود ذو أعداد PRIMARY KEY هو رقم معرف، وغالبا ما يستخدم مع AUTO_INCREMENT.

يجب أن يحتوي كل جدول على عمود مفتاح أساسي (في هذه الحالة: العمود "id"). يجب أن تكون قيمتها فريدة لكل سجل في الجدول.

توضح الأمثلة التالية كيفية إنشاء الجدول في PHP:

مـثـــال (MySQLi Object-oriented):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// انشاء اتصال
$conn = new mysqli($servername, $username,
$password, $dbname);
// تحقق من الاتصال
if ($conn->connect_error) {
    die("Connection failed: " . $conn-
>connect_error);
}

// sql لإنشاء جدول
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY
KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
```

```
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
ON UPDATE CURRENT_TIMESTAMP
)";

if ($conn->query($sql) === TRUE) {
    echo "Table MyGuests created
successfully";
} else {
    echo "Error creating table: " . $conn-
>error;
}

$conn->close();
?>
```

مــثـــال (MySQLi Procedural):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// انشاء اتصال
$conn = mysqli_connect($servername,
$username, $password, $dbname);
// تحقق من الاتصال
if (!$conn) {
    die("Connection failed: " .
```

```
mysqli_connect_error());
}

// لانشاء جدول sql
$sql = "CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY
KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
ON UPDATE CURRENT_TIMESTAMP
)";

if (mysqli_query($conn, $sql)) {
    echo "Table MyGuests created
successfully";
} else {
    echo "Error creating table: " .
mysqli_error($conn);
}

mysqli_close($conn);
?>
```

مــثـــال (PDO):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn
= new PDO("mysql:host=$servername;dbname=$dbn
ame", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

    // sql to create table
    $sql = "CREATE TABLE MyGuests (
    id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY
KEY,
    firstname VARCHAR(30) NOT NULL,
    lastname VARCHAR(30) NOT NULL,
    email VARCHAR(50),
    reg_date TIMESTAMP DEFAULT
CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP
    )";

    // use exec() because no results are
returned
    $conn->exec($sql);
    echo "Table MyGuests created
successfully";
    }
catch(PDOException $e)
```

```
    {
    echo $sql . "<br>" . $e->getMessage();
    }
$conn = null;
?>
```

إدراج البيانات في MySQL

ادراج البيانات في MySQL باستخدام MySQLi وPDO

بعد إنشاء قاعدة بيانات وجدول، يمكننا البدء في إضافة البيانات فيها.

فيما يلي بعض قواعد بناء الجملة التي يجب اتباعها:

- يجب أن يتم الاستعلام عن استعلام SQL في PHP
- يجب أن يتم نقل قيم السلسلة داخل استعلام SQL
- يجب عدم تحديد القيم الرقمية
- يجب ألا تكون الكلمة فارغة

يتم استخدام عبارة INSERT INTO لإضافة سجلات جديدة إلى جدول MySQL:

```
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

ملاحظة: في الفصل السابق، أنشأنا جدولًا فارغًا باسم "MyGuests" بخمسة أعمدة: "id" و "name" و "lastname" و "email" و "reg_date". الآن، دعونا نملأ الجدول بالبيانات.

الأمثلة التالية تضيف سجلا جديدا لجدول "MyGuests":

مــثـــال (MySQLi Object-oriented):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username,
$password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn-
>connect_error);
}
$sql = "INSERT INTO MyGuests (firstname,
lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";
if ($conn->query($sql) === TRUE) {
```

```
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn-
>error;
}
$conn->close();
?>
```

مــثــال (MySQLi Procedural) :

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername,
$username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " .
mysqli_connect_error());
}

$sql = "INSERT INTO MyGuests (firstname,
lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";
```

```
if (mysqli_query($conn, $sql)) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" .
mysqli_error($conn);
}

mysqli_close($conn);
?>
```

مــثـــال (PDO):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";
try {
    $conn
= new PDO("mysql:host=$servername;dbname=$dbn
ame", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
    $sql = "INSERT INTO MyGuests (firstname,
lastname, email)
    VALUES ('John', 'Doe',
'john@example.com')";
    // use exec() because no results are
returned
```

```
     $conn->exec($sql);
     echo "New record created successfully";
     }
catch(PDOException $e)
     {
     echo $sql . "<br>" . $e->getMessage();
     }
$conn = null;
?>
```

الحصول على أخر معرف مدرج في MySQL

الحصول على معرف آخر سجل مدرج

إذا قمنا بإجراء INSERT أو UPDATE على جدول يحتوي على حقل AUTO_INCREMENT، فيمكننا الحصول على معرف آخر سجل تم إدراجه / تحديثه على الفور.

في الجدول "MyGuests" ، العمود "id" هو حقل AUTO_INCREMENT:

```
CREATE TABLE MyGuests (
id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,
firstname VARCHAR(30) NOT NULL,
lastname VARCHAR(30) NOT NULL,
email VARCHAR(50),
reg_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP
)
```

الأمثلة التالية تساوي الأمثلة من الصفحة السابقة (PHP Insert Data Into MySQL)، باستثناء أننا أضفنا سطرًا واحدًا من التعليمات البرمجية لاسترداد معرف السجل الأخير المدرج. نردد نطبع آخر معرّف مدرج:

مــثـــال (MySQLi Object-oriented):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";
// Create connection
$conn = new mysqli($servername, $username,
$password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
$sql = "INSERT INTO MyGuests (firstname,
```

```
lastname, email)
VALUES ('John', 'Doe', 'john@example.com')";
if ($conn->query($sql) === TRUE) {
    $last_id = $conn->insert_id;
    echo "New record created successfully.
Last inserted ID is: " . $last_id;
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}
$conn->close();
?>
```

مـثــال (MySQLi Procedural):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername,
$username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " .
mysqli_connect_error());
}

$sql = "INSERT INTO MyGuests (firstname,
lastname, email)
```

```
VALUES ('John', 'Doe', 'john@example.com')";
if (mysqli_query($conn, $sql)) {
    $last_id = mysqli_insert_id($conn);
    echo "New record created successfully.
Last inserted ID is: " . $last_id;
} else {
    echo "Error: " . $sql . "<br>" .
mysqli_error($conn);
}
mysqli_close($conn);
?>
```

مــثـــال (PDO):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";
try {
    $conn
= new PDO("mysql:host=$servername;dbname=$dbn
ame", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
    $sql = "INSERT INTO MyGuests (firstname,
lastname, email)
    VALUES ('John', 'Doe',
'john@example.com')";
    // use exec() because no results are
```

```
returned
    $conn->exec($sql);
    $last_id = $conn->lastInsertId();
    echo "New record created successfully.
Last inserted ID is: " . $last_id;
    }
catch(PDOException $e)
    {
    echo $sql . "<br>" . $e->getMessage();
    }
$conn = null; ?>
```

أدخل سجلات متعددة في MySQL باستخدام MySQLi و PDO

يجب تنفيذ جمل SQL متعددة مع الدالة `mysqli_multi_query()`.

تضيف الأمثلة التالية ثلاثة سجلات جديدة لجدول "MyGuests":

مــثـــال (MySQLi Object-oriented):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username,
$password, $dbname);
// Check connection
if ($conn->connect_error) {
```

```
    die("Connection failed: " . $conn-
>connect_error);
}

$sql = "INSERT INTO MyGuests (firstname,
lastname, email)
VALUES ('John', 'Doe',
'john@example.com');";
$sql .= "INSERT INTO MyGuests (firstname,
lastname, email)
VALUES ('Mary', 'Moe',
'mary@example.com');";
$sql .= "INSERT INTO MyGuests (firstname,
lastname, email)
VALUES ('Julie', 'Dooley',
'julie@example.com')";

if ($conn->multi_query($sql) === TRUE) {
    echo "New records created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn-
>error;
}

$conn->close();
?>
```

لاحظ أنه يجب فصل كل جمل SQL بفاصلة منقوطة.

مـثـــال (MySQLi Procedural):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername,
$username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " .
mysqli_connect_error());
}

$sql = "INSERT INTO MyGuests (firstname,
lastname, email)
VALUES ('John', 'Doe',
'john@example.com');";
$sql .= "INSERT INTO MyGuests (firstname,
lastname, email)
VALUES ('Mary', 'Moe',
'mary@example.com');";
$sql .= "INSERT INTO MyGuests (firstname,
lastname, email)
VALUES ('Julie', 'Dooley',
'julie@example.com')";

if (mysqli_multi_query($conn, $sql)) {
```

```
    echo "New records created successfully";
} else {
    echo "Error: " . $sql . "<br>" .
mysqli_error($conn);
}

mysqli_close($conn);
?>
```

بطريقة الـ PDO هناك اختلاف بسيط:

مــثـــال (PDO):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn
= new PDO("mysql:host=$servername;dbname=$dbn
ame", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

    // begin the transaction
```

```
    $conn->beginTransaction();
    // our SQL statements
    $conn->exec("INSERT INTO MyGuests
(firstname, lastname, email)
    VALUES ('John', 'Doe',
'john@example.com')");
    $conn->exec("INSERT INTO MyGuests
(firstname, lastname, email)
    VALUES ('Mary', 'Moe',
'mary@example.com')");
    $conn->exec("INSERT INTO MyGuests
(firstname, lastname, email)
    VALUES ('Julie', 'Dooley',
'julie@example.com')");

    // commit the transaction
    $conn->commit();
    echo "New records created successfully";
    }
catch(PDOException $e)
    {
    // roll back the transaction if something
failed
    $conn->rollback();
    echo "Error: " . $e->getMessage();
    }

$conn = null;
?>
```

الجمل المعدة لـ MySQL

الجمل المعدة مهمة جدا وفعاله ضد حقن SQL.

الجمل المعدة والمعلمات المقيدة

الجمل المعدة هي ميزة تستخدم لتنفيذ نفس جمل SQL (أو ما شابه) مرارا وتكرارا بكفاءة عالية.

البيانات المعدة تعمل بشكل أساسي مثل هذا:

١. **إعداد**: يتم إنشاء قالب عبارة SQL وإرسالها إلى قاعدة البيانات. يتم ترك بعض القيم غير سمحددة، وتسمى المعلمات (المسمى "؟"). على سبيل المثال: أدخِل قيم MyGuests (؟ ، ؟ ،؟)
٢. **تقوم قاعدة البيانات بتوزيع وتجميع وتنفيذ** عملية تحسين الاستعلام في قالب بيان SQL، وتخزين النتيجة دون تنفيذها
٣. **التنفيذ**: في وقت لاحق، يقوم التطبيق بربط القيم بالمعلمات، وتنفيذ قاعدة البيانات العبارة. قد يقوم التطبيق بتنفيذ العبارة عدة مرات كما تريد مع قيم مختلفة.

مقارنة بتنفيذ عبارات SQL مباشرة، فإن البيانات المعدة لها ثلاث مزايا رئيسية:

تقلل العبارات المعدة من وقت التحليل حيث يتم التحضير للاستعلام مرة واحدة فقط (على الرغم من تنفيذ العبارة عدة مرات)

تعمل المعلمات Bound على تقليل عرض النطاق الترددي إلى الخادم حيث تحتاج إلى إرسال المعلمات فقط في كل مرة، وليس الاستعلام بالكامل.

تعد العبارات المعدة مفيدة جدًا ضد حقن SQL، لأن قيم المعلمات، التي يتم إرسالها لاحقًا باستخدام بروتوكول مختلف، لا تحتاج إلى الهروب بشكل صحيح. إذا لم يكن قالب البيان الأصلي مشتقًا من الإدخال الخارجي، فلن يحدث حقن SQL.

البيانات المعدة في الخلية

المثال التالي يستخدم البيانات المعدة والمعلمات المنضمة في MySQLi:

مـــثـــال (بالجمل المعدة):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username,
$password, $dbname);

// Check connection
if ($conn->connect_error) {
```

```
    die("Connection failed: " . $conn-
>connect_error);
}

// prepare and bind
$stmt = $conn->prepare("INSERT INTO MyGuests
(firstname, lastname, email) VALUES (?, ?,
?)");
$stmt->bind_param("sss", $firstname,
$lastname, $email);

// set parameters and execute
$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
$stmt->execute();

$firstname = "Mary";
$lastname = "Moe";
$email = "mary@example.com";
$stmt->execute();

$firstname = "Julie";
$lastname = "Dooley";
$email = "julie@example.com";
$stmt->execute();

echo "New records created successfully";

$stmt->close();
$conn->close();
?>
```

أسطر الكود لشرحها من المثال أعلاه:

```
"INSERT INTO MyGuests (firstname, lastname,
email) VALUES (?, ?, ?)"
```

في الـ SQL الخاصة بنا، ندخل علامة السؤال؟ حيث نريد استبدال قيمة عدد صحيح أو سلسلة أو أعداد كسرية أو نقطية.

بعد ذلك، إلقاء نظرة على دالة `bing param()`:

```
$stmt->bind_param("sss", $firstname,
$lastname, $email);
```

هذه الدالة تربط المعلمات مع استعلام SQL وتخبر قاعدة البيانات ما هي المعلمات. يسرد المدخل "sss" أنواع البيانات الموجودة في المعلمات. حرف s يخبر mysql أن المعلمة هي سلسلة.

قد يكون المدخل واحد من أربعة أنواع:

- i – عدد صحيح
- d – كسري (double)
- s – سلسلة
- b – BLOB

يجب أن يكون لدينا واحدة من هذه لكل معلمة.

عن طريق إخبار mysql بنوع البيانات التي نتوقعها، نقوم بتقليل مخاطر حقن SQL.

ملاحظة: إذا أردنا إدراج أي بيانات من مصادر خارجية (مثل إدخال المستخدم)، فمن المهم للغاية أن يتم تعقيم البيانات والتحقق من صحتها.

البيانات المعدة في PDO

المثال التالي يستخدم الجمل المعدة والمعلمات المقيدة في PDO

مـــثـــال (بالجمل المعدة):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn
= new PDO("mysql:host=$servername;dbname=$dbn
ame", $username, $password);
    // set the PDO error mode to exception
```

```
    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

    // prepare sql and bind parameters
    $stmt = $conn->prepare("INSERT INTO
MyGuests (firstname, lastname, email)
    VALUES (:firstname, :lastname, :email)");
    $stmt->bindParam(':firstname',
$firstname);
    $stmt->bindParam(':lastname', $lastname);
    $stmt->bindParam(':email', $email);

    // insert a row
    $firstname = "John";
    $lastname = "Doe";
    $email = "john@example.com";
    $stmt->execute();

    // insert another row
    $firstname = "Mary";
    $lastname = "Moe";
    $email = "mary@example.com";
    $stmt->execute();

    // insert another row
    $firstname = "Julie";
    $lastname = "Dooley";
    $email = "julie@example.com";
    $stmt->execute();

    echo "New records created successfully";
    }
```

```
catch(PDOException $e)
    {
    echo "Error: " . $e->getMessage();
    }
$conn = null;
?>
```

اختيار بيانات MySQL

اختيار بيانات من قواعد بيانات MySQL

جملة الإختيار SELECT تستخدم لاختيار بيانات من جدول واحد أو أكثر:

```
SELECT column_name(s) FROM table_name
```

أو يمكننا إستعمال الحرف * لاختيار كل الأعمدة من الجدول:

```
SELECT * FROM table_name
```

اختيار بيانات MySQLi

المثال التالي يحدد أعمدة المعرّف واسم المستخدم واسم العائلة من جدول MyGuests ويعرضه على الصفحة:

مـثـــال (MySQLi موجهة نحو الكائن):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username,
$password, $dbname);
// Check connection
```

```
if ($conn->connect_error) {
    die("Connection failed: " . $conn-
>connect_error);
}

$sql = "SELECT id, firstname, lastname FROM
MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name:
" . $row["firstname"]. " " .
$row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}
$conn->close();
?>
```

أسطر الكود لشرحها من المثال أعلاه:

أولاً، قمنا بإعداد استعلام SQL الذي يحدد أعمدة المعرِّف واسم المستخدم واسم العائلة من جدول MyGuests. السطر التالي من التعليمات البرمجية يقوم بتشغيل الاستعلام ويضع البيانات الناتجة في متغير يسمى `$result`.

ثم، تقوم الدالة `num_rows()` بالتحقق مما إذا كان هناك أكثر من صفر من الصفوف التي يتم إرجاعها.

إذا كان هناك أكثر من صفر من الصفوف التي يتم إرجاعها، فإن الدالة `fetch_assoc()` تضع كل النتائج في مصفوفة مقترنة التي يمكننا تطبيق الحلقة داخلها. حلقات الحلقة `while()` من خلال مجموعة النتائج وإخراج البيانات من أعمدة المعرف واسم العائلة واسم العائلة.

يوضح المثال التالي نفس المثال أعلاه، بطريقة MySQLi الإجرائية:

مـثـــال (MySQLi Procedural):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername,
$username, $password, $dbname);
// Check connection
if (!$conn) {
```

```
    die("Connection failed: " .
mysqli_connect_error());
}

$sql = "SELECT id, firstname, lastname FROM
MyGuests";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
    // output data of each row
    while($row =
mysqli_fetch_assoc($result)) {
        echo "id: " . $row["id"]. " - Name:
" . $row["firstname"]. " " .
$row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}

mysqli_close($conn);
?>
```

يمكنك أيضا وضع الناتج في جدول HTML:

مــثـــال (MySQLi Procedural):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username,
$password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn-
>connect_error);
}
```

```
$sql = "SELECT id, firstname, lastname FROM
MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    echo "<table><tr><th>ID</th><th>Name</th>
</tr>";
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "<tr><td>".$row["id"]."</td><td>
".$row["firstname"]."
".$row["lastname"]."</td></tr>";
    }
    echo "</table>";
} else {
    echo "0 results";
}
$conn->close();
?>
```

اختيار البيانات مع PDO (+ الجمل المعدة)

المثال التالي يستخدم الجمل المعدة.

تقوم بتحديد أعمدة المعرف والاسم المستعار واسم العائلة من جدول MyGuests وتعرضه في جدول HTML:

مــثــال (PDO):

```
<?php
echo "<table style='border: solid 1px
black;'>";
echo "<tr><th>Id</th><th>Firstname</th><th>La
stname</th></tr>";
class TableRows extends RecursiveIteratorIter
ator {
    function __construct($it) {
```

```
        parent::__construct($it,
self::LEAVES_ONLY);
    }

    function current() {
        return "<td
style='width:150px;border:1px solid
black;'>" . parent::current(). "</td>";
    }

    function beginChildren() {
        echo "<tr>";
    }

    function endChildren() {
        echo "</tr>" . "\n";
    }
}
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";
try {
    $conn
= new PDO("mysql:host=$servername;dbname=$dbn
ame", $username, $password);
    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
    $stmt = $conn->prepare("SELECT id,
firstname, lastname FROM MyGuests");
    $stmt->execute();
    // set the resulting array to associative
```

```
    $result = $stmt-
>setFetchMode(PDO::FETCH_ASSOC);
    foreach(new TableRows(new RecursiveArrayI
terator($stmt->fetchAll())) as $k=>$v) {
        echo $v;
    }
}
catch(PDOException $e) {
    echo "Error: " . $e->getMessage();
}
$conn = null;
echo "</table>";
?>
```

MySQL استخدام جملة WHERE

تحديد وترشيح البيانات من قواعد بيانات MySQL

جملة WHERE تستخدم لتصفية السجلات.

يتم استخدام جملة WHERE لاستخراج تلك السجلات التي تحقق شرطا محددا فقط.

```
SELECT column_name(s) FROM table_name WHERE
column_name operator value
```

تحديد وترشيح البيانات بـ MySQL

المثال التالي يحدد أعمدة الـ id والإسم الأول واسم العائلة من جدول MyGuests حيث اسم العائلة هو "Doe"، ويعرضها على الصفحة:

مـثـــال (MySQLi Object-oriented):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username,
$password, $dbname);
// Check connection
```

```
if ($conn->connect_error) {
    die("Connection failed: " . $conn-
>connect_error);
}

$sql = "SELECT id, firstname, lastname FROM
MyGuests WHERE lastname='Doe'";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name:
" . $row["firstname"]. " " .
$row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}
$conn->close();
?>
```

شرحها أسطر الكود السابقة:

أولاً، قمنا بإعداد استعلام SQL الذي يحدد أعمدة المعرّف واسم المستخدم واسم العائلة من جدول MyGuests حيث يكون اسم العائلة هو "Doe". السطر التالي من التعليمات البرمجية يقوم بتشغيل الاستعلام ويضع البيانات الناتجة في متغير يسمى `$result`.

ثم، تقوم الدالة `num_rows()` بالتحقق مما إذا كان هناك أكثر من صفر من الصفوف التي يتم إرجاعها.

إذا كان هناك أكثر من صفر من الصفوف التي يتم إرجاعها، فإن الدالة `fetch_assoc()` تضع كل النتائج في مصفوفة مقترنة يمكننا استعمال الحلقات فيها. حلقات الحلقة `while()` من خلال مجموعة النتائج وإخراج البيانات من أعمدة id واسم الأول واسم العائلة.

يوضح المثال التالي نفس المثال أعلاه، بطريقة MySQLi الإجرائية:

مـــثـــال (MySQLi Procedural):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername,
$username, $password, $dbname);
// Check connection
if (!$conn) {
```

```
    die("Connection failed: " .
mysqli_connect_error());
}

$sql = "SELECT id, firstname, lastname FROM
MyGuests WHERE lastname='Doe'";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
    // output data of each row
    while($row =
mysqli_fetch_assoc($result)) {
        echo "id: " . $row["id"]. " - Name:
" . $row["firstname"]. " " .
$row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}

mysqli_close($conn);
?>
```

يمكنك أيضا وضع الناتج في جدول HTML:

مـثـــال (MySQLi Object-oriented):

```
<?php
$servername = "localhost";
```

```
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username,
$password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn-
>connect_error);
}

$sql = "SELECT id, firstname, lastname FROM
MyGuests WHERE lastname='Doe'";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    echo "<table><tr><th>ID</th><th>Name</th>
</tr>";
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "<tr><td>".$row["id"]."</td><td>
".$row["firstname"]."
".$row["lastname"]."</td></tr>";
    }
    echo "</table>";
} else {
    echo "0 results";
}
$conn->close();
?>
```

تحديد البيانات بـ PDO (+ الجمل المعدة)

في المثال التالي سنستخدم الجمل المعدة.

وهي تحدد أعمدة id واسم المستخدم واسم العائلة من جدول MyGuests حيث يكون اسم العائلة هو "Doe"، ويعرضها في جدول HTML:

مــثـــال (PDO):

```
<?php
echo "<table style='border: solid 1px
black;'>";
echo "<tr><th>Id</th><th>Firstname</th><th>La
stname</th></tr>";

class TableRows extends RecursiveIteratorIter
```

```
ator {
    function __construct($it) {
        parent::__construct($it,
self::LEAVES_ONLY);
    }

    function current() {
        return "<td
style='width:150px;border:1px solid
black;'>" . parent::current(). "</td>";
    }

    function beginChildren() {
        echo "<tr>";
    }

    function endChildren() {
        echo "</tr>" . "\n";
    }
}
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";
try {
    $conn
= new PDO("mysql:host=$servername;dbname=$dbn
ame", $username, $password);
    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
    $stmt = $conn->prepare("SELECT id,
firstname, lastname FROM MyGuests WHERE
```

```
lastname='Doe'");
    $stmt->execute();
    // set the resulting array to associative
    $result = $stmt-
>setFetchMode(PDO::FETCH_ASSOC);
    foreach(new TableRows(new RecursiveArrayI
terator($stmt->fetchAll())) as $k=>$v) {
        echo $v;
    }
}
catch(PDOException $e) {
    echo "Error: " . $e->getMessage();
}
$conn = null;
echo "</table>";  ?>
```

MySQL باستخدام جملة ORDER BY

تحديد وفرز البيانات من قواعد بيانات MySQL

هذه الجملة تستخدم لفرز النتيجة في ترتيب تصاعدي أو تنازلي.

هذه الجملة تفرز السجلات بترتيب تصاعدي بشكل افتراضي. لفرز السجلات بالترتيب التنازلي، نستخدم الكلمة الأساسية DESC.

```
SELECT column_name(s) FROM table_name ORDER BY
column_name(s) ASC|DESC
```

تحديد وفرز البيانات من MySQLi

المثال التالي يحدد أعمدة الـ id والاسم الأول واسم العائلة من جدول MyGuests. سيتم ترتيب السجلات حسب عمود اسم العائلة:

مـــثـــال (MySQLi Object-oriented):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username,
$password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn-
>connect_error);
}

$sql = "SELECT id, firstname, lastname FROM
MyGuests ORDER BY lastname";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name:
" . $row["firstname"]. " " .
```

```
$row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
}
$conn->close();
?>
```

شرح أسطر الأكواد السابقة:

أولاً، قمنا بإعداد استعلام SQL الذي يحدد أعمدة id واسم المستخدم واسم العائلة من جدول MyGuests. سيتم ترتيب السجلات حسب عمود اسم العائلة. السطر التالي من التعليمات البرمجية يقوم بتشغيل الاستعلام ويضع البيانات الناتجة في متغير يسمى `$result`.

ثم، تقوم الدالة `num_rows()` بالتحقق مما إذا كان هناك أكثر من صفر من الصفوف التي يتم إرجاعها.

إذا كان هناك أكثر من صفر من الصفوف التي يتم إرجاعها، فإن الدالة `fetch_assoc()` تضع كل النتائج في مصفوفة مقترنة يمكننا كتابة حلقة فيها. حلقات الحلقة `while()` من خلال مجموعة النتائج وإخراج البيانات من أعمدة المعرف واسم العائلة واسم العائلة.

يوضح المثال التالي نفس المثال أعلاه، بطريقة MySQLi الإجرائية:

مـثـــال (MySQLi Object-oriented):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername,
$username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " .
mysqli_connect_error());
}

$sql = "SELECT id, firstname, lastname FROM
MyGuests ORDER BY lastname";
$result = mysqli_query($conn, $sql);

if (mysqli_num_rows($result) > 0) {
    // output data of each row
    while($row =
mysqli_fetch_assoc($result)) {
        echo "id: " . $row["id"]. " - Name:
" . $row["firstname"]. " " .
$row["lastname"]. "<br>";
    }
} else {
    echo "0 results";
```

```
}

mysqli_close($conn);
?>
```

يمكنك أيضا وضع النتائج في جدول HTML:

مــثـــال (MySQLi Object-oriented):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
```

```
$conn = new mysqli($servername, $username,
$password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn-
>connect_error);
}

$sql = "SELECT id, firstname, lastname FROM
MyGuests ORDER BY lastname";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    echo "<table><tr><th>ID</th><th>Name</th>
</tr>";
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "<tr><td>".$row["id"]."</td><td>
".$row["firstname"]."
".$row["lastname"]."</td></tr>";
    }
    echo "</table>";
} else {
    echo "0 results";
}
$conn->close();
?>
```

تحديد البيانات بـ PDO (+ الجمل المعدة)

المثال التالي يستخدم الجمل المعدة.

نختار هنا أعمدة المعرِّف واسم المستخدم واسم العائلة من جدول MyGuests. سيتم ترتيب السجلات حسب عمود اسم العائلة، وسيتم عرضها في جدول HTML:

مــثـــال (PDO):

```
<?php
echo "<table style='border: solid 1px
black;'>";
echo "<tr><th>Id</th><th>Firstname</th><th>La
stname</th></tr>";

class TableRows extends RecursiveIteratorIter
ator {
    function __construct($it) {
        parent::__construct($it,
self::LEAVES_ONLY);
    }

    function current() {
        return "<td
style='width:150px;border:1px solid
black;'>" . parent::current(). "</td>";
    }

    function beginChildren() {
        echo "<tr>";
    }
```

```
    function endChildren() {
        echo "</tr>" . "\n";
    }
}

$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn
= new PDO("mysql:host=$servername;dbname=$dbn
ame", $username, $password);
    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
    $stmt = $conn->prepare("SELECT id,
firstname, lastname FROM MyGuests ORDER BY
lastname");
    $stmt->execute();

    // set the resulting array to associative
    $result = $stmt-
>setFetchMode(PDO::FETCH_ASSOC);
    foreach(new TableRows(new RecursiveArrayI
terator($stmt->fetchAll())) as $k=>$v) {
        echo $v;
    }
}
catch(PDOException $e) {
    echo "Error: " . $e->getMessage();
}
```

```
$conn = null;
echo "</table>";
?>
```

MySQL حذف البيانات

حذف البيانات من جدول MySQL باستخدام MySQLi و PDO

جملة DELETE تستخدم لحذف السجلات من الجدول:

```
DELETE FROM table_name
WHERE some_column = some_value
```

ملاحظة: جملة WHERE في بناء جملة DELETE: تحدد جملة WHERE السجل أو السجلات التي يجب حذفها. إذا حذفت جملة WHERE، فسيتم حذف جميع السجلات!

لنلقي نظرة على جدول "MyGuests":

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30
3	Julie	Dooley	julie@example.com	2014-10-26 10:48:23

المثال التالي سيحذف سجل id=3 في جدول "MyGuests":

مـــثـــال (MySQLi Object-oriented):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username,
```

```
$password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn-
>connect_error);
}

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

if ($conn->query($sql) === TRUE) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " . $conn-
>error;
}
$conn->close();
?>
```

مـثـــال (MySQLi Procedural):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername,
```

```
$username, $password, $dbname);
// Check connection
if (!$conn) {
    die("Connection failed: " .
mysqli_connect_error());
}

// sql to delete a record
$sql = "DELETE FROM MyGuests WHERE id=3";

if (mysqli_query($conn, $sql)) {
    echo "Record deleted successfully";
} else {
    echo "Error deleting record: " .
mysqli_error($conn);
}

mysqli_close($conn);
?>
```

مــثـــال (PDO):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn
= new PDO("mysql:host=$servername;dbname=$dbn
```

```
ame", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

    // sql to delete a record
    $sql = "DELETE FROM MyGuests WHERE id=3";

    // use exec() because no results are
returned
    $conn->exec($sql);
    echo "Record deleted successfully";
    }
catch(PDOException $e)
    {
    echo $sql . "<br>" . $e->getMessage();
    }

$conn = null;
?>
```

بعد حذف السجلات، الجدول سيبدو كالتالي:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30

تحديث بيانات MySQL

تحديث البيانات في جدول MySQL باستخدام MySQLi و PDO

يتم استخدام جملة UPDATE لتحديث السجلات الموجودة في جدول:

```
UPDATE table_name
SET column1=value, column2=value2,...
WHERE some_column=some_value
```

ملاحظة: جملة WHERE في بناء جملة UPDATE: تحدد جملة WHERE السجل أو السجلات التي يجب تحديثها. إذا حذفت جملةWHERE، فسيتم تحديث جميع السجلات!

الجدول سيبدو كالتالي:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30

الأمثلة التالية تحدث سجل id=2 في الجدول:

مـــثـــال (MySQLi Object-oriented):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username,
```

```
$password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn-
>connect_error);
}

$sql = "UPDATE MyGuests SET lastname='Doe'
WHERE id=2";

if ($conn->query($sql) === TRUE) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " . $conn-
>error;
}

$conn->close();
?>
```

مـثـــال (MySQLi Procedural):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = mysqli_connect($servername,
$username, $password, $dbname);
// Check connection
```

```
if (!$conn) {
    die("Connection failed: " .
mysqli_connect_error());
}

$sql = "UPDATE MyGuests SET lastname='Doe'
WHERE id=2";

if (mysqli_query($conn, $sql)) {
    echo "Record updated successfully";
} else {
    echo "Error updating record: " .
mysqli_error($conn);
}

mysqli_close($conn);
?>
```

مــثـــال (MySQLi PDO):

```
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDBPDO";

try {
    $conn
```

```
= new PDO("mysql:host=$servername;dbname=$dbn
ame", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);

    $sql = "UPDATE MyGuests SET
lastname='Doe' WHERE id=2";

    // Prepare statement
    $stmt = $conn->prepare($sql);

    // execute the query
    $stmt->execute();

    // echo a message to say the UPDATE
succeeded
    echo $stmt->rowCount() . " records
UPDATED successfully";
    }
catch(PDOException $e)
    {
    echo $sql . "<br>" . $e->getMessage();
    }

$conn = null;
?>
```

بعد تحديث السجل، سيبدو الجدول كما يلي:

id	firstname	lastname	email	reg_date
1	John	Doe	john@example.com	2014-10-22 14:26:15
2	Mary	Moe	mary@example.com	2014-10-23 10:22:30

تحديد اختيارات البيانات MySQL

تحديد اختيارات البيانات من قاعدة بيانات MySQL

يوفر MySQL جملة `LIMIT` التي يتم استخدامها لتحديد عدد السجلات المراد إرجاعها.

تُسهل جملة `LIMIT` برمجة نتائج الصفحات المتعددة أو ترقيم الصفحات مع SQL، وهي مفيدة جدًا في الجداول الكبيرة. إرجاع عدد كبير من السجلات يمكن أن يؤثر على الأداء.

افترض أننا نرغب في تحديد جميع السجلات من 1 إلى 30 (ضمنيًا) من جدول يسمى "الطلبات". سيبدو استعلام SQL كالتالي:

```
$sql = "SELECT * FROM Orders LIMIT 30";
```

عند تشغيل استعلام SQL السابق، فسوف يُرجع السجلات الثلاثين الأولى.

ماذا لو أردنا تحديد السجلات من 16 إلى 25 (ضمناً)؟

يوفر Mysql أيضًا طريقة للتعامل مع هذا: باستخدام `OFFSET`.

يوضح استعلام SQL التالي "إرجاع 10 سجلات فقط، ابدأ في السجل 16 (OFFSET 15)":

```
$sql = "SELECT * FROM Orders LIMIT 10 OFFSET 15";
```

يمكنك أيضا استخدام بنية جملة أقصر لتحقيق نفس النتيجة:

```
$sql = "SELECT * FROM Orders LIMIT 15, 10";
```

لاحظ أنه يتم عكس الأرقام عند استخدام الفاصلة.